

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
1 April 2004 (01.04.2004)

PCT

(10) International Publication Number
WO 2004/028142 A3

(51) International Patent Classification⁷: H04N 7/26

(21) International Application Number:
PCT/YU2003/000027

(22) International Filing Date:
5 September 2003 (05.09.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
p-696/02 17 September 2002 (17.09.2002) YU

(71) Applicants and

(72) Inventors: CEPERKOVIC, Vladimir [YU/YU]; Zelenina gora 27/3, yu-36000 Kraljevo (YU). PAVLOVIC, Sasa [YU/YU]; Kursulina 5, yu-11000 Beograd (YU). MIRKOVIC, Dusan [YU/YU]; 27 Marta 2, yu-11000 Beograd (YU).

(74) Agent: NIKOLIC, Jasna; Cingrijina 20, YU-11000 Beograd (YU).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

Declarations under Rule 4.17:

- as to the identity of the inventor (Rule 4.17(i)) for all designations
- of inventorship (Rule 4.17(iv)) for US only

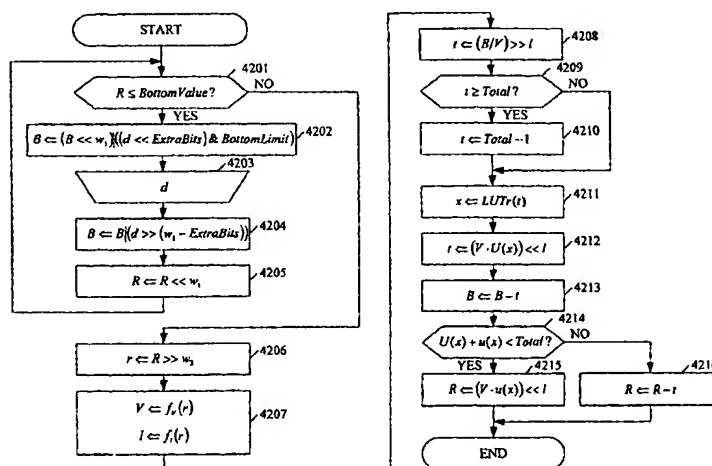
Published:

- with international search report

(88) Date of publication of the international search report:
10 September 2004

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: FAST CODEC WITH HIGH COMPRESSION RATIO AND MINIMUM REQUIRED RESOURCES



(57) Abstract: This invention provides a novel single-pass and multi-pass synchronized encoder and decoder, performing order(s) of magnitude faster data compression and decompression, at any compression ratio with the higher or the same perceived and measured decompressed image quality in comparison with the best state-of-the-art compression methods, using order(s) of magnitude less system resources (processor complexity, memory size, consumed power, bus bandwidth, data latency). These features are achieved using novel direct and inverse non-stationary filters for the recursive octave direct and inverse subband transformation, novel simple context modeling and symbol probability estimation using a minimum number of histograms with the fast adaptation for the sign and the magnitude of the transformation coefficients, a novel accelerated range coder without division operations, and a novel synchronisation of the compressed data.

**FAST CODEC WITH HIGH COMPRESSION RATIO
AND MINIMUM REQUIRED RESOURCES**

5 Technical Field

This invention generally relates to the methods and apparatus for data compression and decompression, and more particularly still and moving image lossless and lossy compression and decompression.

State-of-the-art data compression methods with high compression ratio are time consuming
10 and require complex processors and large memory with high bus bandwidth, which increase consumed power.

The presented technical problem is solved by this invention, which provides order(s) of magnitude faster data compression and decompression, at any compression ratio with the higher or the same perceived and measured decompressed image quality in comparison with the best
15 state-of-the-art compression methods, using order(s) of magnitude less system resources, due to the application of novel direct and inverse non-stationary filters, novel simple context modeling and symbol probability estimation using a minimum number of histograms with the fast adaptation, a novel accelerated range coder without division operations, and a novel synchronization of the compressed data.

20

Background Art

Several tutorials and overviews of still image compression methods are available: O. Egger et al., "High performance compression of visual information – A tutorial review – Part I: Still pictures," *Proc. IEEE*, Vol. 87, No. 6, pp. 976-1011, June 1999; S. Wong et al., "Radiologic
25 image compression – A review," *Proc. IEEE*, Vol. 83, No. 2, pp. 194-219, Feb. 1995; N. D. Memon et al., "Lossless image compression: A comparative study," *Proc. SPIE*, Vol. 2418, pp. 8-20, 1995; and T. Q. Nguyen, "A tutorial on filter banks and wavelets," University of Wisconsin, Madison, WI53706, USA.

A good model of a natural image is based on a power spectrum proportional to f^{-2} , with f
30 being the frequency. This means that most of the energy is concentrated in low-frequency regions. Therefore, a suitable partitioning of the frequency should be finer in low-frequency regions and coarser in high-frequency regions.

For most types of images, direct coding using an entropy coder does not achieve satisfactory compression ratio, so some form of prior decomposition is necessary. Decomposition methods used for still image compression are: predictive, block and subband transformations. Predictive methods are suited for lossless and low compression ratio applications. The main drawbacks of block transformation methods, like discrete cosine transform (DCT), are blocking artifacts at high compression ratios, which are especially visible in image regions with low local variance. Unfortunately, human visual system is very sensitive to such type of image distortion. Subband transformation methods are applicable to both lossless and lossy compression, while the only visible artifact at high compression ratios is the Gibbs phenomenon of linear filters, so-called "ringing effect", described in O. Egger et al., "Subband coding of images using asymmetrical filter banks," *IEEE Trans. Image Processing*, Vol. 4, No. 4, pp. 478-485, Apr. 1995. Due to abundant literature on image compression, the background of this invention is limited to subband transformation.

The subband transformation coefficients are computed by recursively filtering first an input image and then subsequent resulted images with a set of lowpass and highpass filters and down-sampling results. Each subband is separately coded with a bit rate that matches the visual importance of the subband. This leads to visually pleasing image reconstruction and does not produce blocking artifacts. Subband encoding consists of the following four steps: (1) subband decomposition; (2) quantization; (3) probability estimation; and (4) entropy coding of the subbands. The decoding process requires the reversal order of the reversal steps.

The concept of subband transformation was first introduced for speech coding by R. E. Crochiere et al., "Digital coding of speech in subbands," *Bell Syst. Tech. J.*, Vol. 55, No. 8, pp. 1069-1085, Oct. 1976; and U.S. Patent Reg. No. 4,048,443 issued Sep. 1977 to R. E. Crochiere et al. The non-perfect reconstruction filter with a linear phase is two-band QMF, introduced by J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Denver, CO, pp. 291-294, Apr. 9-11, 1980.

The perfect reconstruction filter banks for one-dimensional (1-D) subband transformations were investigated by several authors, like: M. J. Smith et al., "A procedure for designing exact reconstruction filter banks for tree structured subband coders," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, San Diego, CA, pp. 27.1.1-27.1.4, Mar. 1984; T. A. Ramstad, "Analysis/synthesis filter banks with critical sampling," *Proc. Int. Conf. Digital Signal Processing*, Florence, Italy, Sep. 1984; M. Vetterli, "Filter banks allowing perfect

reconstruction," *Signal Processing*, Vol. 10, No. 3, pp. 219-244, Apr. 1986; M. J. Smith et al., "Exact reconstruction techniques for tree structured subband coders," *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. 34, No. 3, pp. 434-441, June 1986; P. P. Vaidyanathan, "Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having perfect reconstruction property," *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. 35, No. 4, pp. 476-496, Apr. 1987; P. P. Vaidyanathan, "Quadrature mirror filter bank, M-band extensions and perfect reconstruction technique," *IEEE Acoustics, Speech, Signal Processing Mag.*, Vol. 4, No. 7, pp. 1035-1037, July 1987; and M. Vetterli et al., "Perfect reconstruction FIR filter banks: Some properties and factorization," *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. 37, No. 7, pp. 1057-1071, July 1989. A design technique leading to numerically perfect reconstruction filter banks has been developed by Nayebi et al., "Time domain filter bank analysis: A new design theory," *IEEE Trans. Signal Processing*, Vol. 40, No. 6, pp. 1412-1429, June 1992. However, such filters are relatively long and thus unsuitable for image-coding applications.

1-D subband transformation theory was extended to two-dimensional (2-D) case by P. J. Burt et al., "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, Vol. 31, No. 4, pp. 532-540, Apr. 1983; M. Vetterli, "Multi-dimensional subband coding: Some theory and algorithms," *Signal Processing*, Vol. 6, No. 2, pp. 97-112, Apr. 1984; J. Woods et al., "Subband coding of images," *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. 34, No. 5, pp. 1278-1288, Oct. 1986; U.S. Patent Reg. No. 4,817,182 issued Mar. 1989 to E. H. Adelson et al., which utilizes 2-D separable QMF banks; A. Zandi et al., "CREW lossless/lossy medical image compression," Ricoh California Research Center, Menlo Park, CA94025, USA, Sep. 12, 1995; and U.S. Patent Reg. No. 6,195,465 issued Feb. 2001 to A. Zandi et al.

State-of-the-art compression algorithms can be divided into three basic groups: single-pass, two-pass and multi-pass. Single-pass algorithms encode/decode image using single access to each transformation coefficient in the memory, as disclosed in C. Chrysafis et al., "Efficient context-based entropy coding for lossy wavelet image compression," *Data Compression Conf.*, Snowbird, UT, Mar. 25-27, 1997. These algorithms are usually limited to prior statistical model with fixed parameters, which typically leads to lower compression ratio than achieved by other methods.

Two-pass algorithms encode/decode image using two accesses to each transformation coefficient in the memory. Therefore, they can use prior statistical model with variable

parameters, which leads to better compression ratio than in the single-pass case. However, they need to store all transformation coefficients in the memory, in order to perform second pass, which requires additional memory size of the order of an uncompressed input image.

Multi-pass algorithms encode/decode image based on an implicitly defined static model
5 (JPEG2000, SPIHT and EZW). JPEG2000 was described in C. Christopoulos et al. "The JPEG2000 still image coding system: An overview," *IEEE Trans. Consum. Electr.*, Vol. 46, No. 4, pp. 1103-1127, Nov. 2000. Set partitioning in hierarchical trees (SPIHT) algorithm was disclosed in A. Said et al., "Image compression using the spatial-orientation tree," *Proc. IEEE Int. Symp. Circuits Systems*, Chicago, IL, pp. 279-282, May 1993; A. Said et al., "A new fast and
10 efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Tech.*, Vol. 6, No. 3, pp. 243-250, June 1996; and U.S. Patent Reg. No. 5,764,807 issued June 1998 to W. A. Pearlman et al. Alphabet and group partitioning of transformation coefficients was disclosed in U.S. Patent Reg. No. 5,959,560 issued Sep. 1999 to A. Said et al. Embedded Zerotrees Wavelet (EZW) algorithm was described in J. M. Shapiro, "Embedded
15 image coding using zerotrees of wavelets coefficients," *IEEE Trans. Signal Processing*, Vol. 41, No. 12, pp. 3445-3462, Dec. 1993. EZW technique is based on: (1) partial ordering of the transformation coefficients by magnitude using a set of octavely decreasing thresholds; (2) transmission of order by a subset partitioning algorithm that is duplicated at the decoder; (3) ordered bit plane transmission of refinement bits; and (4) utilization of the self-similarity of the
20 transformation coefficients across different subbands. The Embedded Predictive Wavelet Image Coder (EPWIC), based on the conditional probability model in addition to EZW, was disclosed in R. W. Buccigrossi et al., "Progressive wavelet image coding based on a conditional probability model," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Munich, Germany, Vol. 4, pp. 2597-2600, Apr. 21-24, 1997; and E. P. Simoncelli et al., "Progressive wavelet image
25 compression using linear inter-band magnitude prediction," *Proc. 4th Int. Conf. Image Processing*, Santa Barbara, CA, Oct. 26-29, 1997. All these methods store the complete image in the memory and require relatively large number of passes to encode/decode image.

A number of authors have observed that subband transformation coefficients have highly non-Gaussian statistics, according to B. A. Olshausen et al., "Natural image statistics and
30 efficient coding," *Network: Computation in Neural Systems*, Vol. 7, No. 2, pp. 333-339, July 1996; R. W. Buccigrossi et al., "Image compression via joint statistical characterization in the wavelet domain," GRASP Laboratory Technical Report #414, University of Pennsylvania, USA,

30 May 1997; E. P. Simoncelli et al., "Embedded wavelet image compression based on a joint probability model," *Proc. 4th Int. Conf. Image Processing*, Santa Barbara, CA, USA, Oct. 26-29, 1997; and R. W. Buccigrossi et al., "Image compression via joint statistical characterization in the wavelet domain," *IEEE Trans. Image Processing*, Vol. 8, No. 12, pp. 1688-1701, Dec. 1999.

5 The reason is a spatial structure of typical images consisting of smooth regions interspersed with sharp edges. The smooth regions produce near-zero transformation coefficients, while the sharp edges produce large-amplitude transformation coefficients. Statistics of transformation coefficients can be modeled by two-parameter "generalized Laplacian" density function, which sharply peaks at zero, with more extensive tails compared to a Gaussian density function, as in S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Analysis Machine Intelligence*, Vol. 11, No. 7, pp. 674-693, July 1989; and E. P. Simoncelli et al., "Noise removal via bayesian wavelet coring," *Proc. 3rd Int. Conf. Image Processing*, Lausanne, Switzerland, Vol. 1, pp. 379-383, Sep. 1996. Unfortunately, two-pass algorithm is necessary for the calculation of density function parameters. Furthermore, 15 experimental results show significant disagreement between this density function and actual histograms at higher levels of subband transformation. The lowpass subband contains almost entirely positive transformation coefficients, appropriate to uniform density function.

Higher compression ratio can be achieved by defining symbols based on the context model, i.e. on the basis of neighborhood transformation coefficients, analogously to text compression 20 methods. An analysis of zero-tree and other wavelet coefficient context models was described in S. Todd et al., "Parameter reduction and context selection for compression of gray-scale images," *IBM J. Res. Develop.*, Vol. 29, No. 2, pp. 188-193, Mar. 1985; V. R. Algazi et al., "Analysis based coding of image transform and subband coefficients," *SPIE Applications of Digital Image Processing XVIII*, Vol. 2564, pp. 11-21, July 1995; S. D. Stearns, "Arithmetic coding in lossless waveform compression," *IEEE Trans. Signal Processing*, Vol. 43, No. 8, pp. 1874-1879, Aug. 25 1995; and U.S. Patent Reg. No. 6,222,941 issued Apr. 2001 to A. Zandi et al.

It is possible to find a bit code, which is more efficient than the fixed-length code, if the probability of an occurrence of a particular symbol is known. Codeword assignment is usually done by variable-length coding, run-length coding, Huffman coding and arithmetic coding. 30 Techniques for removing alphabetical redundancy mostly generate prefix codes, and mostly transform the messages into a bit string, assigning longer codes to less probable symbols, as in B. M. Oliver et al., "Efficient coding," *Bell Syst. Tech. J.*, Vol. 31, No. 4, pp. 724-750, July 1952;

D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, Vol. 40, No. 9, pp. 1098-1101, Sep. 1952; and E. N. Gilbert et al., "Variable length binary encodings," *Bell Syst. Tech. J.*, Vol. 38, No. 4, pp. 933-967, July 1959.

The highest compression ratio is achieved by arithmetic coding, which theoretically can
5 remove all redundant information from a digitized message, according to L. H. Witten et al.,
"Arithmetic coding for data compression," *Commun. ACM*, Vol. 30, No. 6, pp. 520-540, June
1987; A. Moffat et al., "Arithmetic coding revisited," *Proc. Data Compression Conf.*, Snowbird,
UT, pp. 202-211, Mar. 1995; and A. Moffat et al., "Arithmetic coding revisited," *ACM Trans.*
Inform. Syst., Vol. 16, No. 3, pp. 256-294, July 1998.

10 Arithmetic Q-coder is disclosed in Mitchell et al., "Software implementations of the Q-
coder," *IBM J. Res. Develop.*, Vol. 21, No. 6, pp. 753-774, Nov. 1988; W. B. Pennebaker et al.,
"An overview of the basic principles of the Q-coder adaptive binary arithmetic coder," *IBM J.*
Res. Develop., Vol. 32, No. 6, pp. 717-726, Nov. 1988; and U.S. Patent Reg. Nos.: 4,933,883
issued June 1990; and 4,935,882 issued June 1990 to W. B. Pennebaker et al.

15 Arithmetic Z-coder is disclosed in L. Bottou et al., "The Z-coder adaptive coder," *Proc. Data*
Compression Conf., Snowbird, UT, pp. 13-22, Mar. 1998; and U.S. Patent Reg. Nos.: 6,188,334
issued Feb. 2001; 6,225,925 issued May 2001; and 6,281,817 issued Aug. 2001 to Y. Bengio et
al.

However, this invention is based on the range coder disclosed in G. N. N. Martin, "Range
20 encoding: an algorithm for removing redundancy from a digitised message," *Proc. Video & Data*
Recording Conf., Southampton, UK, July 24-27, 1979.

Both processing time and memory size of state-of-the-art lossy image compression methods
increase with the compression ratio. State-of-the-art microprocessors, signal processors and even
microcontrollers have small capacity of fast memory (general-purpose processor registers and
25 internal or external cache memory), and large capacity of several times slower memory (external
system memory). This invention fits most or even all necessary temporary data into this fast
memory, thus additionally achieving fastest algorithm execution.

The common approach for decreasing the required memory size is to divide the large image
into blocks and encode each block independently. All best state-of-the-art still image
30 compression methods (JPEG2000, JPEG, etc.) and moving image compression methods (MPEG-
4, MPEG-2, MPEG-1, etc.) are block based, as described in D. Santa-Cruz et al., "JPEG2000 still

image coding versus other standards," *Proc. SPIE 45th annual meeting, Applications of Digital Image Processing XXIII*, San Diego, CA, Vol. 4115, pp. 446-454, July 30 - Aug. 4, 2000.

JPEG2000 encoder first divides an input uncompressed image into non-overlapping blocks, then recursively subband transforms each block independently using direct discrete wavelet transform (DWT), according to M. Boliek et al. (editors), "JPEG2000 Part I Final Draft International Standard," (ISO/IEC FDIS15444-1), ISO/IEC JTC1/SC29/WG1 N1855, Aug. 18, 2000. The transformation coefficients are then quantized and entropy coded, before forming the output codestream. The input codestream in decoder is first entropy decoded, dequantized and recursively subband transformed into independent blocks using inverse DWT, in order to produce the reconstructed image. However, tiling produces blocking artifacts at the boundaries between blocks. This drawback can be partially eliminated by framing, i.e. overlapping of the neighborhood blocks for at least one pixel. Another serious drawback is quality degradation at higher compression ratios and limited maximum acceptable compression ratio.

JPEG2000 standard supports two filtering modes: a convolution and a lifting. The signal should be first extended periodically on both ends for half-length of the filter. Convolution-based filtering consists of performing a series of multiplications between the low-pass and high-pass filter coefficients and samples of the extended 1-D signal. Lifting-based filtering consists of a sequence of alternative updating of odd sample values of the signal with a weighted sum of even sample values, and updating of even sample values with a weighted sum of odd sample values, according to W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmonic Analysis*, Vol. 3, No. 2, pp. 186-200, 1996; and W. Sweldens, "The lifting scheme: Construction of second generation wavelets," *SIAM J. Math. Anal.*, Vol. 29, No. 2, pp. 511-546, 1997.

JPEG2000 utilizes MQ arithmetic coder similar to the QM coder adopted in the original JPEG standard described by G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, Vol. 38, No. 1, pp. 18-34, Feb. 1992; U.S. Patent Reg. Nos.: 5,059,976 issued Oct. 1991; and 5,307,062 issued Apr. 1994 to F. Ono et al.

JPEG standard was described in "Digital compression and coding of continuous-tone still images," Int. Org. Standardization ISO/IEC, JTC1 Committee Draft, JPEG 8-R8, 1990; and G. K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, Vol. 34, No. 4, pp. 30-44, Apr. 1991. The original image is divided into 8×8 blocks, which are separately transformed using DCT. After transformation, the 64 transform coefficients are quantized by different

quantization steps in order to account the different importance of each transformation coefficient, using smaller quantization steps for low-frequency coefficients than those for high-frequency coefficients. The transformation coefficients are then coded using either Huffman or arithmetic coding. The independent quantization of blocks causes blocking effect. JPEG lossless
5 compression does not use transform, but a prediction for the removal of a redundant information between neighboring pixels. The prediction error is coded by a Huffman code. The compression ratio is about 2:1 for natural images.

MPEG-4 video compression standard is object based and is developed to compress sequences of images using interframe coding. However, its intraframe coding is a still image
10 compression method, very similar to JPEG. The bounding box of the object to be coded is divided into macroblocks of 16×16 size, containing four blocks of 8×8 pixels for the luminance and two blocks of 8×8 pixels for the down-sampled chrominance. DCT is performed separately on each of the blocks in a macroblock, coefficients are quantized, zigzag scanned, and entropy coded by run-length and Huffman methods.

15 Disclosure of the Invention

The first object of this invention is to provide novel single-pass and multi-pass synchronized encoder and decoder, performing order(s) of magnitude faster data compression and decompression, at any compression ratio with the higher or the same perceived and measured
20 decompressed image quality in comparison with the best state-of-the-art compression methods (JPEG2000, JPEG, MPEG-4, MPEG-2, MPEG-1, etc.), using order(s) of magnitude less system resources (processor complexity, memory size, consumed power, bus bandwidth, data latency).

The second object of this invention is to provide novel direct and inverse non-stationary filters for the recursive octave direct and inverse subband transformation (pyramidal
25 decomposition), in order to fulfill the first object of this invention.

The third object of this invention is to provide novel simple context modeling and symbol probability estimation using a minimum number of histograms with fast adaptation for the sign and the magnitude of the transformation coefficients, provided by the second object of this invention, in order to fulfill the first object of this invention.

30 The fourth object of this invention is to provide novel accelerated range coder without division operations, due to the utilization of the context models and symbol probability

estimation provided by the third object of this invention, in order to fulfill the first object of this invention.

The fifth object of this invention is to provide a novel synchronization of the compressed data provided by the fourth object of this invention, in order to fulfill the first object of this invention.

5 All objects of this invention may be implemented either in hardware or software or their combination. The input uncompressed data can be still or video images, audio, digitized analog data, executable code, text or any digital data.

Brief Description of Drawings

10 The advantages and features of this invention will be readily apparent to those skilled in the art from the detailed description of the preferred embodiments of this invention in connection with the accompanying drawings in which:

FIG. 1 is a block diagram of state-of-the-art communication system employing compression and decompression.

15 **FIG. 2** is a block diagram of state-of-the-art encoder for the compression.

FIG. 3 is a block diagram of state-of-the-art decoder for the decompression.

FIG. 4 illustrates results during three levels of direct two-dimensional subband transformation (direct 2DST) of an input uncompressed image, as well as during three levels of inverse 2DST, producing an output uncompressed image.

20 **FIG. 5** and **FIG. 6** are block diagrams of state-of-the-art low-memory encoder and the decoder, respectively, with three levels of one-dimensional subband transformation (1DST).

FIG. 7 and **FIG. 8** are block diagrams of state-of-the-art low-memory encoder and the decoder, respectively, with three levels of 2DST.

25 **FIG. 9** and **FIG. 10** are block diagrams of the first embodiment of the encoder and the decoder of this invention, respectively, with three levels of 1DST.

FIG. 11 and **FIG. 12** are block diagrams of the first embodiment of the encoder and the decoder of this invention, respectively, with three levels of 2DST.

FIG. 13 and **FIG. 14** are block diagrams of the second embodiment of the encoder and the decoder of this invention, respectively, with three levels of 1DST.

30 **FIG. 15** and **FIG. 16** are block diagrams of the second embodiment of the encoder and the decoder of this invention, respectively, with three levels of 2DST.

FIG. 17 is a block diagram of the general non-stationary filter cell of this invention, used in all embodiments of the direct and inverse non-stationary filters.

FIG. 18 is a block diagram of the general integer-to-integer non-stationary filter cell of this invention, used in all embodiments of the direct and inverse non-stationary filters.

5 **FIG. 19** and **FIG. 20** are block diagrams of all embodiments of the direct and inverse non-stationary filter of this invention, respectively.

FIG. 21 and **FIG. 22** are block diagrams of the first embodiment of the direct and inverse non-stationary filter of this invention, respectively.

10 **FIG. 23** and **FIG. 24** are block diagrams of the second embodiment of the direct and inverse non-stationary filter of this invention, respectively.

FIG. 25 and **FIG. 26** illustrate transfer function in the frequency domain of the second embodiment of the direct and inverse non-stationary filter of this invention, respectively.

FIG. 27 and **FIG. 28** are block diagrams of the third embodiment of the direct and inverse non-stationary filter of this invention, respectively.

15 **FIG. 29** and **FIG. 30** are flowcharts of a single level 2DST using first embodiment of direct and inverse non-stationary filters of this invention, respectively.

FIG. 31 is a flowchart of the encoding probability estimator and entropy encoder of this invention based on single-pass adaptive histograms.

FIG. 32 shows the magnitude-sets for the context modeling.

20 **FIG. 33** shows the signs for the context modeling.

FIG. 34 is a flowchart of the entropy decoder and the decoding probability estimator of this invention based on single-pass adaptive histograms.

FIG. 35 is an initialization flowchart for histograms with fast adaptation.

FIG. 36 is an update flowchart for histogram with fast adaptation.

25 **FIG. 37A** illustrates one-dimensional probability function example with instantaneous jump and instantaneous fall, used for testing of histograms with fast adaptation.

FIG. 37B illustrates one-dimensional estimated probability function, produced by the state-of-the-art probability estimation algorithm.

30 **FIG. 37C** illustrates one-dimensional estimated probability function, produced by the first embodiment of the probability estimation algorithm of this invention, but without fast adaptation of histograms.

FIG. 37D illustrates one-dimensional estimated probability function, produced by the second embodiment of the probability estimation algorithm of this invention, with fast adaptation of histograms.

FIG. 37E illustrates close-up of one-dimensional estimated probability function, produced by the second embodiment of the probability estimation algorithm of this invention, with fast adaptation of histograms.

FIG. 38 is a flowchart of the state-of-the-art *flush* procedure.

FIG. 39 and FIG. 40 are flowcharts of the state-of-the-art range encoder and the range decoder, respectively.

FIG. 41A,B and FIG. 42 are flowcharts of the range encoder and the range decoder of this invention, without division operations and optionally, without multiplication operations.

Best Mode(s) for Carrying Out the Invention

FIG. 1 is a block diagram of state-of-the-art communication system employing the compression of the input uncompressed image 10 inside the encoder 30, in order to fill the output compressed buffer 32 with the output compressed image 18, which is transmitted through the transmission channel 34, in order to fill the input compressed buffer 33 with the input compressed image 19, which is decompressed inside the decoder 31, in order to produce the output uncompressed image 11. The input uncompressed image 10 is preferably color or gray-scale image in YUV 4:4:4 or 4:2:2 formats. However, any other input image format, like RGB or YCrCb is also applicable using appropriate color space converter for input image format conversion into YUV formats, which is well known to that skilled-in-the-art.

FIG. 2 is a block diagram of the state-of-the-art encoder 30. The input uncompressed image 10 is received by the direct subband transformer 20. The output of the direct subband transformer 20 are transformation coefficients 12, which can be quantized into the quantized transformation coefficients 14 in the quantizer 24, in case of lossy compression, or just passed to the encoding probability estimator 26, in case of lossless compression. The outputs of the encoding probability estimator 26 are symbol probabilities 16 within the specified contexts, which are used by the entropy encoder 28, in order to produce the output compressed image 18.

FIG. 3 is a block diagram of state-of-the-art decoder 31. The input compressed image 19 is received by the entropy decoder 29. The output 15 of the entropy decoder 29 is received by the decoding probability estimator 27, in order to reconstruct the symbol probabilities 17 within the

specified contexts, and feed them back into the entropy decoder 29. The output 15 of the entropy decoder 29 is also received by the dequantizer 25, in order to produce dequantized transformation coefficients 13, in case of lossy compression, or just passed to the inverse subband transformer 21, in case of lossless compression. The output of the inverse subband transformer 21 is the output uncompressed image 11.

FIG. 4 illustrates results during three level state-of-the-art direct and inverse two-dimensional subband transformation. The number of levels is usually fixed and is between three and seven in state-of-the-art systems. However, in this invention, the number of subband transformation levels N is variable and depends on the image size, according to the:

$$N = \lceil \log_2 (\min(W, H) / K_S) \rceil,$$

where W is the image width (number of pixels in each line), H is the image height (number of lines), parameter K_S is preferably 17, and brackets $\lceil \rceil$ denote ceil function, i.e. a minimum integer number higher than the floating point number within the brackets.

State-of-the-art single-level 1DST is performed by low-pass filtering input data and downsampling by two, in order to produce the subband L, and high-pass filtering input data and downsampling by two, in order to obtain the subband H.

State-of-the-art multi-level 1DST is performed by applying single-level 1DST onto the input uncompressed data in the level 0, and then subsequently performing single-level 1DST onto the subband L_i , produced as a result of previous single-level 1DST. Level 0 subbands are L_0 and H_0 . Level 1 subbands L_1 and H_1 are produced by applying low-pass and high-pass filters onto the subband L_0 , respectively. Level 2 subbands L_2 and H_2 are produced by applying low-pass and high-pass filters onto the subband L_1 , respectively, etc.

State-of-the-art single-level 2DST is performed by applying 1DST separately, first horizontally along the rows and then vertically along the columns. The results of a single-level 2DST are four subbands: LL, LH, HL and HH.

The subband LL corresponds to the low-pass filtering along the rows and the low-pass filtering along the columns, and contains simultaneously low frequency horizontal information and low frequency vertical information. Most of the typical image energy is concentrated in this subband.

The subband LH corresponds to the low-pass filtering along the rows and the high-pass filtering along the columns, and contains simultaneously low frequency horizontal information and high frequency vertical information, i.e. horizontal edge information.

The subband HL corresponds to the high-pass filtering along the rows and the low-pass filtering along the columns, and contains simultaneously high frequency horizontal information and low frequency vertical information, i.e. vertical edge information.

5 The subband HH corresponds to the high-pass filtering along the rows and the high-pass filtering along the columns, and contains simultaneously high frequency horizontal information and high frequency vertical information, i.e. diagonal edge information.

State-of-the-art multi-level 2DST is performed by applying single-level 2DST onto the input uncompressed image 10 in the level 0, and then subsequently performing single-level 2DST onto the subband LL_i , produced as a result of previous single-level 2DST. Level 0 subbands are LL_0 , LH_0 , HL_0 and HH_0 . Level 1 subbands LL_1 , LH_1 , HL_1 and HH_1 are produced by applying 2DST onto the subband LL_0 . Level 2 subbands LL_2 , LH_2 , HL_2 and HH_2 are produced by applying 2DST onto the subband LL_1 , etc.

The block diagrams in FIG. 5, FIG. 6, FIG. 7 and FIG. 8 are described in more details in: C. Chrysafis et al., "Line based reduced memory wavelet image compression," *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 30 – Apr. 1, 1998; C. Chrysafis et al., "An algorithm for low memory wavelet image compression," *Proc. IEEE Int. Conf. Image Processing (ICIP)*, Kobe, Japan, 24-28 Oct. 1999; C. Chrysafis et al., "Line based reduced memory wavelet image compression," *IEEE Trans. Image Processing*, Vol. 9, No. 3, pp. 378-389, Mar. 2000; C. Chrysafis, "Wavelet image compression rate distortion optimizations and complexity reductions," *Ph.D. Thesis*, University of Southern California, USA, Mar. 2000; and C. Chrysafis et al., "Minimum memory implementations of the lifting scheme," *SPIE, Int. Symp. Optical Science Tech.*, San Diego, CA, July 30-Aug. 4, 2000.

C. Chrysafis' papers describe line-based approach, in which an input uncompressed image 10 is read line by line. The order in which subband transformation coefficients are generated by the direct 2DST is almost opposite to the order expected by the inverse 2DST, thus requiring additional synchronization memory between the encoder and the decoder. The total synchronization memory size is usually equally split between the encoder and the decoder. However, it is possible to assign the complete synchronization memory to encoder only or to decoder only, depending on the particular application. Further description will be devoted only to the symmetrical synchronization memory of equal size in the encoder and the decoder.

C. Chrysafis first considered the memory size for 1DST based on the convolution filtering with the finite impulse response (FIR) filter with odd filter length $L = 2 \cdot D + 1$ and symmetric

extension of the input uncompressed data at the boundaries. The samples are received serially with period T_p . After delay of $D \cdot T_p$, the total of $L = D + 1 + D$ samples were received due to the extension, so the filter starts generation of high-pass subband H_0 coefficients with $2 \cdot T_p$ period, as well as the generation of low-pass subband L_0 coefficients with $2 \cdot T_p$ period, due to the downsampling by two.

Unfortunately, linear phase FIR filters are computationally expensive in terms of both required processing time and memory. These drawbacks are eliminated by the computationally efficient direct and inverse non-stationary filters of this invention, which will be described further. The memory needed for filtering depends on D and a size of the internal variable used for the storing operation according to TABLE 1.

TABLE 1

| Filtering method | Filtering memory size F per 1DST level [internal variable] | Total filtering memory size for N 1DST levels [internal variable] |
|--|--|---|
| Convolution | $2 \cdot D + 1$ | $(2 \cdot D + 1) \cdot N$ |
| Chrysafis' lifting | $D + 2$ | $(D + 2) \cdot N$ |
| Lifting | $D + 1$ | $(D + 1) \cdot N$ |
| Non-stationary filters of this invention | D | $D \cdot N$ |

FIG. 5 and FIG. 6 are block diagrams of state-of-the-art low-memory encoder 30 and the decoder 31 with $N = 3$ levels of direct 1DST (D1DST) and inverse 1DST (I1DST), respectively. The period between receipts of successive samples of the input uncompressed data is T_p . The delay in the level 0 D1DST 100 is $D \cdot T_p$. The period between the generations of successive transformation coefficients in each of subbands L_0 and H_0 is $2 \cdot T_p$.

The delay in the level 1 D1DST 101 is $2 \cdot D \cdot T_p$. This delay is compensated by the synchronization memory z^{-D} 120 of size D for the subband H_0 , due to the period $2 \cdot T_p$ between the generations of successive transformation coefficients. The period between the generations of successive transformation coefficients in each of subbands L_1 and H_1 is $4 \cdot T_p$.

The delay in the level 2 D1DST 102 is $4 \cdot D \cdot T_p$. This delay is compensated by the synchronization memory $z^{-2 \cdot D}$ 121 of size $2 \cdot D$ for the subband H_0 , due to the period $2 \cdot T_p$ between the generation of successive transformation coefficients, as well as the synchronization

memory z^{-D} 122 of size D for the subband H_1 , due to the period $4 \cdot T_p$ between the generations of successive transformation coefficients. The period between the generations of successive transformation coefficients in each of subbands L_2 and H_2 is $8 \cdot T_p$.

Thanks to the compensation of delays, all transformation coefficients become available at the same time for the quantization in quantizers 140-143, encoding probability estimation in encoding probability estimators 160-163 and entropy encoding in entropy encoders 180-183. However, in the actual implementation, quantization can be performed before the storage in the synchronization memories 120-122, in order to decrease the necessary total synchronization memory size, especially if 32-bit floating point transformation coefficients are the result of irreversible DWT using 9-tap/7-tap (9/7) filter, disclosed in M. Antonini et al., "Image coding using the wavelet transform," *IEEE Trans. Image Proc.*, Vol. 1, No. 2, pp. 205-220, April 1992, and applied in JPEG2000 standard.

The symmetrical situation is valid for the decoder 31, requiring their own synchronization memories 130-132. The synchronization memory sizes for the encoder 30 or the decoder 31 with $N = 3$ 1DST levels are given in TABLE 2.

TABLE 2

| 1DST level | Input data period | 1DST level delay | Output data period | 1DST level 1 synchronization memory size | 1DST level 2 synchronization memory size | Total synchronization memory size |
|------------|-------------------|-----------------------|--------------------|--|--|-----------------------------------|
| 0 | T_p | $D \cdot T_p$ | $2 \cdot T_p$ | D | $2 \cdot D$ | $D + 2 \cdot D = 3 \cdot D$ |
| 1 | $2 \cdot T_p$ | $2 \cdot D \cdot T_p$ | $4 \cdot T_p$ | - | D | D |
| 2 | $4 \cdot T_p$ | $4 \cdot D \cdot T_p$ | $8 \cdot T_p$ | - | - | 0 |
| All | - | - | - | - | - | $3 \cdot D + D = 4 \cdot D$ |

The synchronization memory sizes in transformation coefficient units, for the encoder 30 or the decoder 31 within the communication system utilizing N 1DST levels are given in TABLE 3. As the number N of 1DST levels increases, the synchronization memory size for all 1DST levels increases much faster than the filtering memory size.

FIG. 7 and FIG. 8 are block diagrams of state-of-the-art low-memory encoder 30 and the decoder 31, with $N = 3$ levels of direct 2DST (D2DST) and inverse 2DST (I2DST), respectively. After the level 0 1DST horizontal filtering is finished, the complete lines of transformation coefficients are stored in the memory, requiring memory size W for each line, where W is the number of pixels within each line, i.e. the width of the image. The width of the

subband image is reduced by two with each increment of 2DST level. Therefore, each subband LL_{i+1} at 2DST level $i+1$ requires memory for half of the lines of the previous subband LL_i at 2DST level i , according to TABLE 4.

5

TABLE 3

| 1DST level | Synchronization memory size per 1DST level [coefficient] |
|------------|--|
| 0 | $D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-2} \cdot D = \sum_{k=0}^{N-2} 2^k \cdot D = (2^{N-1} - 1) \cdot D$ |
| 1 | $D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-3} \cdot D = \sum_{k=0}^{N-3} 2^k \cdot D = (2^{N-2} - 1) \cdot D$ |
| 2 | $D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-4} \cdot D = \sum_{k=0}^{N-4} 2^k \cdot D = (2^{N-3} - 1) \cdot D$ |
| n | $D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-n-2} \cdot D = \sum_{k=0}^{N-n-2} 2^k \cdot D = (2^{N-n-1} - 1) \cdot D$ |
| $N-3$ | $D + 2 \cdot D = 3 \cdot D$ |
| $N-2$ | D |
| All levels | $\sum_{n=0}^{N-2} (2^{N-n-1} - 1) \cdot D = (2^N - N - 1) \cdot D$ |

The period between receipts of successive lines of the input uncompressed image 10 is T_L . The delay in the level 0 D2DST 200 is $D \cdot T_L$. The period between the generations of successive lines in the subband LL_0 is $2 \cdot T_L$ and the width of each line is $W/2$. The delay in the level 1 D2DST 201 is $2 \cdot D \cdot T_L$. This delay is compensated by the synchronization memories z^{-D} 220-222 of size $D \cdot W/2$ for each of the subbands LH_0 , HL_0 and HH_0 , due to the period $2 \cdot T_L$ between the generations of successive lines and the width $W/2$ of each line. The period between the generations of successive lines in each of subbands LL_1 , LH_1 , HL_1 and HH_1 is $4 \cdot T_L$ and the width of each line is $W/4$.

15 The delay in the level 2 D2DST 202 is $4 \cdot D \cdot T_L$. This delay is compensated by the synchronization memories $z^{-2 \cdot D}$ 223-225 of size $2 \cdot D \cdot W/2$ for each of the subbands LH_0 , HL_0 and HH_0 , due to the period $2 \cdot T_L$ between the generations of successive lines and the width $W/2$ of each line, as well as the synchronization memories z^{-D} 226-228 of size $D \cdot W/4$ for each of the subbands LH_1 , HL_1 and HH_1 , due to the period $4 \cdot T_L$ between the generations of successive

lines and the width $W/4$ of each line. The period between the generations of successive lines in each of subbands LL_2 , LH_2 , HL_2 and HH_2 is $8 \cdot T_L$.

TABLE 4

| 2DST level | Filtering memory size per 2DST level [internal variables] |
|------------|---|
| 0 | $F \cdot W$ |
| 1 | $\frac{F \cdot W}{2}$ |
| 2 | $\frac{F \cdot W}{4}$ |
| n | $\frac{F \cdot W}{2^n}$ |
| $N-1$ | $\frac{F \cdot W}{2^{N-1}}$ |
| All levels | $\sum_{n=0}^{N-1} \frac{F \cdot W}{2^n} = 2 \cdot F \cdot W \cdot \left(1 - \frac{1}{2^N}\right) < 2 \cdot F \cdot W$ |

5 Thanks to the compensation of delays, all transformation coefficients become available at the same time for the quantization in quantizers 240-249, encoding probability estimation in encoding probability estimators 260-269 and entropy encoding in entropy encoders 280-289. However, in the actual implementation, quantization can be performed before storage in the synchronization memories 220-228, in order to decrease the necessary total synchronization
10 memory size.

The symmetrical situation is valid for the decoder 31, requiring their own synchronization memories 230-238. The synchronization memory sizes in coefficient units, for the encoder 30 or the decoder 31 with $N = 3$ 2DST levels is given in TABLE 5.

15 The synchronization memory sizes in transformation coefficient units, for the encoder 30 or the decoder 31 with N 2DST levels is given in TABLE 6. It should be noticed that the last two results from TABLE 6 are correct, in comparison with erroneous equation (5.1) at page 76 (page 90 in PDF file) of C. Chrysafis' Ph.D. Thesis. The total synchronization memory size is derived by multiplying the last result from TABLE 6, with the number of bytes in a single transformation coefficient, or more exactly the number of bytes in a single quantized transformation coefficient,
20 if quantization is performed before buffering of transformation coefficients.

TABLE 5

| 2DST level | Input line period | 2DST level delay | Output line period | 2DST level 1 synchronization memory size | 2DST level 2 synchronization memory size | Total synchronization memory size |
|------------|-------------------|-----------------------|--------------------|--|--|-----------------------------------|
| 0 | T_L | $D \cdot T_L$ | $2 \cdot T_L$ | $D \cdot W/2$ | $2 \cdot D \cdot W/2$ | $3 \cdot D \cdot W/2$ |
| 1 | $2 \cdot T_L$ | $2 \cdot D \cdot T_L$ | $4 \cdot T_L$ | - | $D \cdot W/4$ | $D \cdot W/4$ |
| 2 | $4 \cdot T_L$ | $4 \cdot D \cdot T_L$ | $8 \cdot T_L$ | - | - | 0 |
| All | - | - | - | - | - | $7 \cdot D \cdot W/4$ |

FIG. 9 and FIG. 10 are block diagrams of the first embodiment of the encoder 30 and the decoder 31 of this invention, with $N = 3$ levels of D1DST and I1DST, respectively.

TABLE 6

| 2DST level | Total synchronization memory size per subband [coefficient] Synchronization memory size for 3 subbands (LH, HL and HH) is 3 times higher |
|-----------------------------|--|
| 0 | $(D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-2} \cdot D) \cdot \frac{W}{2} = \frac{W}{2} \cdot \sum_{k=0}^{N-2} 2^k \cdot D = (2^{N-1} - 1) \cdot D \cdot \frac{W}{2}$ |
| 1 | $(D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-3} \cdot D) \cdot \frac{W}{4} = \frac{W}{4} \cdot \sum_{k=0}^{N-3} 2^k \cdot D = (2^{N-2} - 1) \cdot D \cdot \frac{W}{4}$ |
| 2 | $(D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-4} \cdot D) \cdot \frac{W}{8} = \frac{W}{8} \cdot \sum_{k=0}^{N-4} 2^k \cdot D = (2^{N-3} - 1) \cdot D \cdot \frac{W}{8}$ |
| n | $(D + 2 \cdot D + 2^2 \cdot D + \dots + 2^{N-n-2} \cdot D) \cdot \frac{W}{2^{n+1}} = \frac{W}{2^{n+1}} \cdot \sum_{k=0}^{N-n-2} 2^k \cdot D = (2^{N-n-1} - 1) \cdot D \cdot \frac{W}{2^{n+1}}$ |
| $N-3$ | $(D + 2 \cdot D) \cdot \frac{W}{2^{N-2}} = 3 \cdot D \cdot \frac{W}{2^{N-2}}$ |
| $N-2$ | $D \cdot \frac{W}{2^{N-1}}$ |
| All levels | $\sum_{n=0}^{N-2} (2^{N-n-1} - 1) \cdot D \cdot \frac{W}{2^{n+1}} = \frac{1}{3} \cdot \left(2^N - 3 + \frac{1}{2^{N-1}} \right) \cdot D \cdot W$ |
| All levels and all subbands | $3 \cdot \sum_{n=0}^{N-2} (2^{N-n-1} - 1) \cdot D \cdot \frac{W}{2^{n+1}} = \left(2^N - 3 + \frac{1}{2^{N-1}} \right) \cdot D \cdot W$ |

Instead of buffering the transformation coefficients in the synchronization memory of the encoder 30, input uncompressed data are direct subband transformed in D1DSTs 100-102, using direct non-stationary filters, transformation coefficients are quantized in quantizers 140-143, the probabilities of transformation coefficients within the specified contexts are estimated in the

encoding probability estimators 160-163, the quantized transformation coefficients are entropy encoded in the entropy encoders 180-183 and finally in their compressed form pass through the synchronization memories 320-321, in order to produce the output compressed data, which are temporarily stored into the output compressed buffer 32, from which they will be transmitted.

5 Instead of buffering the transformation coefficients in the synchronization memory of the decoder 31, input compressed data, received from the input compressed buffer 33, pass through the synchronization memories 330-331, then are decoded in the entropy decoders 190-193, with the help of the decoding probability estimators 170-173, after that the quantized transformation coefficients are dequantized in the dequantizers 150-153, and finally inverse subband
10 transformed in I1DSTs 110-112, using inverse non-stationary filters, and stored in their uncompressed form in the output uncompressed data memory.

FIG. 11 and FIG. 12 are block diagrams of the first embodiment of the encoder 30 and the decoder 31 of this invention, with $N = 3$ levels of D2DST and I2DST, respectively.

15 Instead of buffering the transformation coefficients in the synchronization memory of the encoder 30, input uncompressed image 10 is direct subband transformed in D2DSTs 200-202, using direct non-stationary filters, transformation coefficients are quantized in quantizers 240-249, the probabilities of transformation coefficients within the specified contexts are estimated in the encoding probability estimators 260-269, the quantized transformation coefficients are entropy encoded in the entropy encoders 280-289 and finally in their compressed form pass
20 through the synchronization memories 420-425, in order to produce the output compressed image 18, which is temporarily stored into the output compressed buffer 32, from which it will be transmitted.

25 Instead of buffering the transformation coefficients in the synchronization memory of the decoder 31, the input compressed image 19, received from the input compressed buffer 33, passes through the synchronization memories 430-435, then is decoded in the entropy decoders 290-299, with the help of the decoding probability estimators 270-279, after that the quantized transformation coefficients are dequantized in the dequantizers 250-259, and finally inverse subband transformed in I2DSTs 210-212, using inverse non-stationary filters, and stored in their uncompressed form in the output uncompressed image 11 memory.

30 Therefore, the first embodiment of this invention requires order of magnitude less memory size, proportional to the compression ratio CR , in comparison with the results given in TABLE 3 in page 92 (page 106 of PDF file) of C. Chrysafis' Ph.D. Thesis, and several orders of magnitude

less memory size in comparison with best state-of-the-art communication systems employing compression (JPEG2000, JPEG, MPEG-4, MPEG-2 and MPEG-1). Furthermore, the column filtering is performed as soon as the sufficient number of lines has been horizontally filtered (TABLE 1). For example, the first embodiment of the direct non-stationary filter of this invention requires only $D = 2$ horizontal lines in order to start column filtering.

FIG. 13 and **FIG. 14** are block diagrams of the second embodiment of the encoder **30** and the decoder **31** of this invention, with $N = 3$ levels of D1DST and I1DST, respectively.

Instead of buffering the compressed transformation coefficients in the synchronization memory of the encoder **30**, input uncompressed data is direct subband transformed in D1DSTs **100-102**, using direct non-stationary filters, transformation coefficients are quantized in quantizers **140-143**, the probabilities of transformation coefficients within the specified contexts are estimated in the encoding probability estimators **160-163**, the quantized transformation coefficients are entropy encoded in the entropy encoders **180-183** and finally in their compressed form stored into the output compressed buffer **32**, which temporarily serves as the synchronization memory, from which they will be transmitted.

Instead of buffering the compressed transformation coefficients in the synchronization memory of the decoder **31**, input compressed data, received from the input compressed buffer **33**, are decoded in the entropy decoders **190-193**, with the help of the decoding probability estimators **170-173**, after that the quantized transformation coefficients are dequantized in the dequantizers **150-153**, and finally inverse subband transformed in I1DSTs **110-112**, using inverse non-stationary filters, and stored in their uncompressed form in the output uncompressed data memory.

FIG. 15 and **FIG. 16** are block diagrams of the second embodiment of the encoder **30** and the decoder **31** of this invention, with $N = 3$ levels of D2DST and I2DST, respectively.

Instead of buffering the compressed transformation coefficients in the synchronization memory of the encoder **30**, input uncompressed image **10** is direct subband transformed in D2DSTs **200-202**, using direct non-stationary filters, transformation coefficients are quantized in quantizers **240-249**, the probabilities of transformation coefficients within the specified contexts are estimated in the encoding probability estimators **260-269**, the quantized transformation coefficients are entropy encoded in the entropy encoders **280-289** and finally in their compressed form stored into the output compressed buffer **32**, from which the output compressed image **18** will be transmitted.

Instead of buffering the compressed transformation coefficients in the synchronization memory of the decoder 31, input compressed image 19, received from the input compressed buffer 33, is decoded in the entropy decoders 290-299, with the help of the decoding probability estimators 270-279, after that the quantized transformation coefficients are dequantized in the dequantizers 250-259, and finally inverse subband transformed in I2DSTs 210-212, using inverse non-stationary filters, and stored in their uncompressed form in the output uncompressed image 11 memory.

Since neither input uncompressed image 10 memory, nor output compressed image 18 memory are considered as a part of the image compression system, the second embodiment of this invention does not require synchronization memory, according to TABLE 7.

TABLE 7

| Method | Total filter memory size [internal variable] | Total synchronization memory size [coefficient] |
|---|--|---|
| Chrystafis' with convolution | $2 \cdot (2 \cdot D + 1) \cdot W \cdot \left(1 - \frac{1}{2^N}\right)$ | $D \cdot W \cdot \left(2^N - 3 + \frac{1}{2^{N-1}}\right)$ |
| Chrystafis' with lifting | $2 \cdot (D + 2) \cdot W \cdot \left(1 - \frac{1}{2^N}\right)$ | $D \cdot W \cdot \left(2^N - 3 + \frac{1}{2^{N-1}}\right)$ |
| First embodiment of present invention | $2 \cdot D \cdot W \cdot \left(1 - \frac{1}{2^N}\right)$ | $D \cdot W \cdot \left(2^N - 3 + \frac{1}{2^{N-1}}\right) / CR$ |
| Second embodiment of present invention | $2 \cdot D \cdot W \cdot \left(1 - \frac{1}{2^N}\right)$ | 0 |

It should be noticed that neither first nor the second embodiment of this invention can be represented in a block form of FIG. 2 and FIG. 3, due to highly distributed quantizers, encoding probability estimators, entropy encoders, entropy decoders, decoding probability estimators, dequantizers and synchronization memories. However, the generic numbering introduced in FIG. 2 and FIG. 3 will be used in further description for simplicity reasons.

FIG. 17 is a block diagram of the general non-stationary filter cell (NSFC) of this invention, used in all embodiments of the direct and inverse non-stationary filters. NSFC provides octave subband decomposition, while keeping linear phase, contrary to state-of-the-art stationary infinite impulse response (IIR) filters, which are the basis for the invention of NSFC. Additionally, the realization of non-stationary filters is twice simpler than the realization of the equivalent FIR filter. Furthermore, non-stationary filters of this invention provide alternate low-pass and high-

pass filtered result at its output, with even and odd sample indexes, respectively, additionally reducing realization complexity. Total number of delay elements is usually about two times less than in two FIR filters mutually performing the same task (TABLE 7). Finally, the first and the second embodiment of the direct and inverse non-stationary filters of this invention do not utilize multipliers due to the filter coefficients that are powers of two.

General NSFC $F_{N_1, N_2, \bar{K}_1, \bar{K}_2}(x, c)$ has two NSFC inputs, x and c , as well as single NSFC output y . Thereto, x and y represent an input and an output of a filter device 805, which nonstationarity is controlled by an input c through the first switch 800 and the second switch 801. The input samples are received serially, one sample at each period at input x . Input c is, for example, $c = 0$ for the samples with even indexes and $c = 1$ for the samples with odd indexes, although it can be defined in the opposite manner. The output samples with even indexes are low-pass filtered and down-sampled. The output samples with odd indexes are high-pass filtered and down-sampled. The specific embodiments depend on a choice of the parameters N_1 , N_2 , \bar{K}_1 and \bar{K}_2 , where N_1 is a first function 802, N_2 is a second function 803, \bar{K}_1 is the constant vector with elements $K_1[0], K_1[1], \dots, K_1[k-2], K_1[k-1]$, denoted as 601, 603, ..., 600+m-3, 600+m-1, respectively, and \bar{K}_2 is the constant vector with elements $K_2[0], K_2[1], \dots, K_2[k-2], K_2[k-1]$, denoted as 600+m-2, 600+m-4, ..., 602, 600, respectively.

The following description of the general NSFC operation will be described for case of filters with odd number of delay elements z^{-m} 500, 501, ..., 500+m-2, i.e. even $m = 2 \cdot k$. The outputs of even indexed delay elements z^{-m} 500, 502, ..., 500+m-4, 500+m-2 are routed through the multipliers $K_1[0]$ 601, $K_1[1]$ 603, ..., $K_1[k-1]$ 600+m-1, which are substituted by shifters in the first and the second embodiment of the direct and inverse non-stationary filters of this invention. The outputs of the multipliers $K_1[0]$ 601, $K_1[1]$ 603, ..., $K_1[k-1]$ 600+m-1 are summed together using the adders 701, 703, ..., 700+m-3, the sum is transformed in N_1 802, fed back and finally summed with input x in the adder 700 for the closed switch 800, i.e. for $c = 1$. The NSFC input x and the outputs of odd indexed delay elements z^{-m} 501, 503, ..., 500+m-3 are routed through the multipliers $K_2[k-1]$ 600, $K_2[k-2]$ 602, ..., $K_2[1]$ 600+m-4, $K_2[0]$ 600+m-2, which are substituted by shifters in the first and the second embodiment of the direct and inverse non-stationary filters of this invention. The outputs of the multipliers $K_2[k-1]$ 600, $K_2[k-2]$ 602, ..., $K_2[1]$ 602, $K_2[0]$ 600+m-2 are summed together using the adders 702, ..., 700+m-4,

700+m-2, the sum is transformed in N_2 803, fed forward and finally summed with the output of the last delay element z^{-w} 500+m-2 in the adder 700+m-1 for the closed switch 801, i.e. for $c = 0$, in order to produce the NSFC output y .

If the direct NSFC is defined by $F_{N_1, N_2, \bar{K}_1, \bar{K}_2}(x, c)$, the inverse NSFC is defined by
 5 $F_{-N_1, -N_2, \bar{K}_1, \bar{K}_2}(x, \bar{c})$, where $-N_1$ and $-N_2$ represents functions with the negative sign, while \bar{c} represents inverted binary variable c , which is 1 for the samples with even indexes and 0 for the samples with odd indexes. It is obvious that both direct and inverse NSFC have the same structure as the general NSFC. The serial connection of the direct and inverse NSFC provides the transfer function $z^{-2 \cdot w \cdot (m-1)}$, i.e. time delay of $2 \cdot w \cdot (m-1)$ samples, where $m-1$ is the number
 10 of delay elements z^{-w} in each of direct and inverse NSFC. The order of received pixels is usually left to right, and top to bottom. The delay element z^{-w} within the horizontal filter has to store only a single pixel, i.e. $w = 1$. However, the delay element z^{-w} for the vertical filter has to store the complete horizontal line with W pixels, so $w = W$.

Although the preferred embodiment of the general, direct and inverse NSFCs are described
 15 and illustrated, those skilled in the art can perform various modifications and design equivalents, since there is an infinite number of filter cells with the same transfer functions up to the time shift. For example, the time delay of the transfer function will be increased simply by appending arbitrary number of zeros to vectors \bar{K}_1 or \bar{K}_2 . It is also possible to split single NSFC into multiple NSFCs. For example, serial connection of three NSFCs, $F_{N_1, 0, \bar{K}_1, [0]}$, $F_{0, 0, [0], [0]}$ and
 20 $F_{0, N_2, [0], \bar{K}_2}$ provide the time delayed transfer function $F_{N_1, N_2, \bar{K}_1, \bar{K}_2}$, which can be implemented in a single NSFC. Such variations and equivalents should not be regarded as a departure from the spirit and the scope of invented NSFC, since they are obvious to those skilled in the art.

It can be shown that sufficient conditions for providing linear phase in NSFC are
 25 $N_1 = N_2 = \text{const}$ and $\bar{K}_1 = \bar{K}_2$. However, said conditions are not necessary. For example, it can be shown that $F_{1, 1, [1], [-1]}$ also provides linear phase.

FIG. 18 is a block diagram of the general integer-to-integer non-stationary filter cell (INSFC) $FI_{N_1, N_2, \lambda, \xi, \bar{K}_1, \bar{K}_2, K_3}(x, c)$ of this invention, which can be used in all embodiments of the direct and inverse non-stationary filters. General INSFC is based on a general NSFC 860. Newly introduced parameters are binary constant λ , real constant ξ and prescaling factor K_3 850. The definition

of the parameterized nonlinear block $N_{\lambda,\xi}(x,c)$ 870 utilizes the following annotations: $\lfloor w \rfloor$ as the largest integer not exceeding w (i.e. "floor" function), and $\lceil w \rceil$ as the smallest integer greater than w (i.e. "ceil" function).

If the direct INSFC is defined by $FI_{N_1,N_2,\lambda,\xi,\bar{\lambda},\bar{c},K_3}(x,c)$, inverse INSFC is defined by
 5 $FI_{-N_1,-N_2,\bar{\lambda},\bar{\xi},\bar{K}_1,\bar{K}_2,K_3}(x,\bar{c})$, where $-N_1$ and $-N_2$ represents functions with the negative sign, while $\bar{\lambda}$ represents inverted binary constant λ and \bar{c} represents inverted binary variable c . It is obvious that both direct and inverse INSFC have the same structure as the general INSFC.

$$N_{\lambda,\xi}(x,c) = \begin{cases} \frac{x}{K_3}, & c = 1 \\ \left\lfloor \frac{x}{K_3} + \xi \right\rfloor, & c = 0 \wedge \lambda = 0 \\ \left\lceil \frac{x}{K_3} - \xi \right\rceil, & c = 0 \wedge \lambda = 1 \end{cases}$$

FIG. 19 is a block diagram of all embodiments of the direct non-stationary filter of this
 10 invention, which are made of sequentially connected direct NSFCs or direct INSFCs 900, 901,..., 900+e-1. Multipliers with constants G_1 881 and G_2 882 provide unity gain during both low-pass and high-pass direct filtering operation, depending on the position of the switch 880, which passes signal multiplied with constant G_1 for $c=1$, or signal multiplied with constant G_2 for $c=0$. Integer-to-integer transform is performed with $G_1 = G_2 = 1$.

FIG. 20 is a block diagram of all embodiments of the inverse non-stationary filter of this
 15 invention, which are made of sequentially connected inverse NSFCs or inverse INSFCs 950+e-1,..., 951, 950 in the inversed order in comparison with FIG. 19. Multipliers with constants G_1^{-1} 891 and G_2^{-1} 892 provide unity gain during both low-pass and high-pass inverse filtering, depending on the position of the switch 890, which passes signal multiplied with constant G_1^{-1}
 20 for $c=1$, or signal multiplied with constant G_2^{-1} for $c=0$. Integer-to-integer transform is performed with $G_1 = G_2 = 1$.

In a practical implementation one adder could be removed at each connection of two NSFCs. Namely, either a feedback signal from the next NSFC or a feedforward signal from the previous NSFC has zero value within the period of c , so two adders can be replaced by a single adder

with the multiplexed input. This modification is possible even in INSFC. Namely, the nonlinear block $N_{b,\xi}(x,c)$ reduces itself to a gain block when fed forward value is zero.

FIG. 21 is a block diagram of the first embodiment of the direct non-stationary filter of this invention, made of sequentially connected two first-order direct NSFCs $F_1 = F_{-1,-1,[1/2],[1/2]}(x,c)$ 1000 and $F_2 = F_{1,1,[1/4],[1/4]}(x,c)$ 1001. The transfer function of this filter is appropriate to the 5-tap/3-tap analysis (5/3) filter, disclosed in D. Le Gall et al., "Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, New York, NY, pp. 761-765, Apr. 1988, which is used for reversible DWT in JPEG2000 standard.

FIG. 22 is a block diagram of the first embodiment of the inverse non-stationary filter of this invention, made of sequentially connected two first-order inverse NSFCs $F_2^{-1} = F_{-1,-1,[1/4],[1/4]}(x,c)$ 1011 and $F_1^{-1} = F_{1,1,[1/2],[1/2]}(x,c)$ 1010. The transfer function of this filter is appropriate to the 5-tap/3-tap synthesis (5/3) filter, disclosed in D. Le Gall et al., "Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques," *Proc. Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, New York, NY, pp. 761-765, Apr. 1988, which is used for reversible DWT in JPEG2000 standard.

FIG. 23 is a block diagram of the second embodiment of the direct non-stationary filter of this invention, made by the addition of one third-order direct NSFC $F_3 = F_{-1,-1,[1/16] \ -1/16],[1/16] \ -1/16]}(x,c)$ 1002 after the first embodiment of the direct non-stationary filter of this invention. FIG. 24 is a block diagram of the second embodiment of the inverse non-stationary filter of this invention, made by the addition of one third-order inverse NSFC $F_3^{-1} = F_{1,1,[1/16] \ -1/16],[1/16] \ -1/16]}(x,c)$ 1012 before the first embodiment of the inverse non-stationary filter of this invention.

FIG. 25 illustrates transfer function in the frequency domain of the second embodiment of the direct non-stationary filter of this invention, respectively. FIG. 26 illustrates transfer function in the frequency domain of the second embodiment of the inverse non-stationary filter of this invention, respectively. Neither the first nor the second embodiment of the non-stationary filters of this invention utilizes multipliers with constants G_1 881 and G_2 882, so the third embodiment is especially designed to show their use.

Multipliers in both the first and the second embodiment of the non-stationary filters of this inventions can be made either as shifters or shifted hardwired bit line connections (shifted

connections between output and input bit lines) in hardware, or shift instructions or bit remapping data structures (bit fields in C programming language) in software.

FIG. 27 is a block diagram of the third embodiment of the direct non-stationary filter of this invention, made of sequentially connected four first-order direct NSFCs: $F_4 = F_{-1,-1,[\alpha],[\alpha]}(x,c)$ 1020, $F_5 = F_{-1,-1,[\beta],[\beta]}(x,c)$ 1021, $F_6 = F_{1,1,[\gamma],[\gamma]}(x,c)$ 1022 and $F_7 = F_{1,1,[\Delta],[\Delta]}(x,c)$ 1023, with parameters given in the TABLE 8.

FIG. 28 is a block diagram of the third embodiment of the inverse non-stationary filter of this invention, made of sequentially connected four first-order direct NSFCs: $F_7^{-1} = F_{-1,-1,[\Delta],[\Delta]}(x,c)$ 1033, $F_6^{-1} = F_{-1,-1,[\gamma],[\gamma]}(x,c)$ 1032, $F_5^{-1} = F_{1,1,[\beta],[\beta]}(x,c)$ 1031 and $F_4^{-1} = F_{1,1,[\alpha],[\alpha]}(x,c)$ 1030, with parameters given in the TABLE 8.

TABLE 8

| Parameter | Value |
|-----------|------------------|
| α | 1.58193248659365 |
| β | 0.07167834102835 |
| γ | 0.82577375069311 |
| Δ | 0.52307224508739 |
| G_1 | 0.76323962993937 |
| G_2 | 1.31020450298084 |

It should be also noticed that all filter embodiments use a symmetrical extension of input data at the image boundaries. However, it is possible to implement non-stationary filter coefficients near image boundaries, in order to provide the same effect, which is well known to that skilled-in-the-art. Therefore, such unnecessary description will be omitted, since it does not influence the spirit of this invention.

FIG. 29 is a flowchart of a single level 2DST using first embodiment of direct non-stationary filters of this invention in the encoder 30, for odd number of lines as an example. Line index i is initialized in the processing block 2901. Horizontal 1DST with direct non-stationary filters is performed in the processing block 2902, using alternate $c = 0$ for pixels with even indexes and $c = 1$ for pixels with odd indexes. Vertical 1DST with direct non-stationary filter with $c = 0$ is performed in the processing block 2903. The processing of all other lines is performed within the loop until the height H of the input uncompressed image 10 is reached in the examination block 2912. Line index i is incremented in the processing block 2904. Horizontal 1DST with direct

non-stationary filters is performed in the processing block 2905, using alternate $c = 0$ for pixels with even indexes and $c = 1$ for pixels with odd indexes. Vertical 1DST with direct non-stationary filter with $c = 1$ is performed in the processing block 2906. The low-pass transformed coefficients of a single line are output in the output block 2907. Line index i is incremented again
5 in the processing block 2908. Horizontal 1DST with direct non-stationary filters is performed in the processing block 2909, using alternate $c = 0$ for pixels with even indexes and $c = 1$ for pixels with odd indexes. Vertical 1DST with direct non-stationary filter with $c = 0$ is performed in the processing block 2910. The high-pass transformed coefficients of a single line are output in the output block 2911. Vertical 1DST with direct non-stationary filter with $c = 1$ is performed
10 in the processing block 2913. The low-pass transformed coefficients of a single line are output in the output block 2914.

FIG. 30 is a flowchart of a single level 2DST using first embodiment of inverse non-stationary filters of this invention in the decoder 31, for odd number of lines as an example. Line index i is initialized in the processing block 3001. Vertical 1DST with inverse non-stationary
15 filter with $c = 1$ is performed in the processing block 3002. The processing for all other lines is performed within the loop until the height H of the image is reached in the examination block 3011. Line index i is incremented in the processing block 3003. Vertical 1DST with inverse non-stationary filter with $c = 0$ is performed in the processing block 3004. Horizontal 1DST with inverse non-stationary filters is performed in the processing block 3005, using alternate $c = 1$ for
20 pixels with even indexes and $c = 0$ for pixels with odd indexes. The pixels of a single line are output in the output block 3006. Line index i is incremented again in the processing block 3007. Vertical 1DST with inverse non-stationary filter with $c = 1$ is performed in the processing block 3008. Horizontal 1DST with inverse non-stationary filters is performed in the processing block 3009, using alternate $c = 1$ for pixels with even indexes and $c = 0$ for pixels with odd indexes.
25 The pixels of a single line are output in the output block 3010. Vertical 1DST with inverse non-stationary filter with $c = 0$ is performed in the processing block 3012. Horizontal 1DST with inverse non-stationary filters is performed in the processing block 3013, using alternate $c = 1$ for pixels with even indexes and $c = 0$ for pixels with odd indexes. The pixels of a single line are output in the output block 3014.

30 TABLE 9 shows an example of the order of the generation of transformation coefficients within subbands for a hypothetical 8 pixels \times 8 lines image during single-level 2DST in the encoder 30. The output of the direct non-stationary filter generates alternatively low-pass

transformation coefficients with even indexes and high-pass transformation coefficients with odd indexes.

Most state-of-the-art compression methods use the same 1DST filter for both rows and columns, since all directions in the image should be treated equally. However, this decreases computational efficiency. This invention can successfully use different filters for horizontal and vertical filtering. Since horizontal filter has much smaller memory/quality ratio, we may choose better and more complex horizontal filter, and slightly worse and less complex vertical filter, practically without increasing memory size. Three embodiments of the direct and inverse non-stationary filters provide total of nine two-dimensional filters according to TABLE 10.

TABLE 9

| ORDER OF THE GENERATION OF TRANSFORMATION COEFFICIENTS | | | | | | | | | | | | | | | |
|--|----|----|----|------------|----|----|----|------------|----|----|----|------------|----|----|----|
| Subband LL | | | | Subband HL | | | | Subband LH | | | | Subband HH | | | |
| 1 | 3 | 5 | 7 | 2 | 4 | 6 | 8 | 9 | 11 | 13 | 15 | 10 | 12 | 14 | 16 |
| 17 | 19 | 21 | 23 | 18 | 20 | 22 | 24 | 25 | 27 | 29 | 31 | 26 | 28 | 30 | 32 |
| 33 | 35 | 37 | 39 | 34 | 36 | 38 | 40 | 41 | 43 | 45 | 47 | 42 | 44 | 46 | 48 |
| 49 | 51 | 53 | 55 | 50 | 52 | 54 | 56 | 57 | 59 | 61 | 63 | 58 | 60 | 62 | 64 |

TABLE 10

| Vertical filter embodiment | Horizontal filter embodiment | Required memory | Required multipliers |
|----------------------------|------------------------------|-----------------|----------------------|
| 1 | 1 | Low | No |
| 1 | 2 | Low | No |
| 1 | 3 | Low | Yes |
| 2 | 1 | Medium | No |
| 2 | 2 | Medium | No |
| 2 | 3 | Medium | Yes |
| 3 | 1 | High | Yes |
| 3 | 2 | High | Yes |
| 3 | 3 | High | Yes |

Quantization of transformation coefficients is performed in the quantizer 24 only in case of lossy compression. Dequantization of transformation coefficients is performed in the dequantizer 25 only in case of lossy compression. Transformation coefficients directly pass through the quantizer 24 and dequantizer 25 in case of lossless compression.

Quantized or non-quantized transformation coefficients 14 in each subband are encoded separately without any information from any other subband, in order to fulfill the first objection of this invention.

FIG. 31 is a flowchart of the encoding probability estimator 26 and entropy encoder 28 of this invention, based on single-pass adaptive histograms. The adaptation starts from a uniform distribution and requires several samples to complete. The adaptation time is proportional to a number of histogram bins and the difference between the uniform distribution and the exact distribution of the variable to be encoded. Positive and negative transformation coefficients have the same probability in LH, HL and HH subbands, providing opportunity to adapt histograms according to magnitude only, thus halving the total number of unknown probabilities and twice increasing the adaptation speed. The transformation coefficients C , which are input in the input block 3101, are split into sign S and magnitude M pairs in the processing block 3102. This method is also described in A. Said et al., "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Processing*, Vol. 5, No. 9, pp. 1303-1310, Sep. 1996. Sign S takes three values as in C. Chrysafis' Ph.D. Thesis, but with different code.

$$S = \begin{cases} 0, & C > 0 \\ 2, & C = 0 \\ 1, & C < 0 \end{cases}$$

The statistics of transformation coefficients for most images directly leads to the definition of bin boundaries based on the logarithmic progression, as in W. A. Pearlman, "High performance, low complexity image compression," *Applications of Digital Image Processing X, Proc. SPIE* 3164, pp. 234-246, July 1997; and U.S. Pat. No. 5,959,560 issued Sep. 1999 to A. Said et al. However, this invention utilizes higher number of histogram bins (32 in total), than state-of-the-art methods. The logarithms of base 2, used in the processing block 3103 of this invention, define the magnitude-set index MS to be equal to a sum of a doubled position of the first nonzero bit of the highest significance and the value of the first next bit of the lower significance, in a binary representation of the magnitude M , according to TABLE 11, which is given for 16-bit coefficients. A residual R is defined as the difference between the magnitude M and the lower coefficient limit, equal to a value of the magnitude M with all bits zeroed except the first nonzero bit of the highest significance and the first next bit of the lower significance in a binary representation of the magnitude M , according to TABLE 11. Therefore, the transformation coefficients are actually split in the processing block 3104 into sign S , magnitude-set index MS

and residual R triples. Due to this, the probability density distributions of MS and R can be approximately treated as uniform.

TABLE 11

| Coefficient limits (inclusive) | | Coefficient range | MS |
|--------------------------------|-------|-------------------|------|
| Lower | Upper | | |
| 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 2 | 1 | 2 |
| 3 | 3 | 1 | 3 |
| 4 | 5 | 2 | 4 |
| 6 | 7 | 2 | 5 |
| 8 | 11 | 4 | 6 |
| 12 | 15 | 4 | 7 |
| 16 | 23 | 8 | 8 |
| 24 | 31 | 8 | 9 |
| 32 | 47 | 16 | 10 |
| 48 | 63 | 16 | 11 |
| 64 | 95 | 32 | 12 |
| 96 | 127 | 32 | 13 |
| 128 | 191 | 64 | 14 |
| 192 | 255 | 64 | 15 |
| 256 | 383 | 128 | 16 |
| 384 | 511 | 128 | 17 |
| 512 | 767 | 256 | 18 |
| 768 | 1023 | 256 | 19 |
| 1024 | 1535 | 512 | 20 |
| 1536 | 2047 | 512 | 21 |
| 2048 | 3071 | 1024 | 22 |
| 3072 | 4095 | 1024 | 23 |
| 4096 | 6143 | 2048 | 24 |
| 6144 | 8191 | 2048 | 25 |
| 8192 | 12287 | 4096 | 26 |
| 12288 | 16383 | 4096 | 27 |
| 16384 | 24575 | 8192 | 28 |
| 24576 | 32767 | 8192 | 29 |
| 32768 | 49151 | 16384 | 30 |
| 49152 | 65535 | 16384 | 31 |

The predictor for MS based on the local variance is used to improve compression ratio. Since MS value can be considered as an approximation of a logarithm, a logarithm of a local variance (proportional to an entropy) can be calculated as a mean value \overline{MS} in the processing block 3106 from neighborhood magnitude-set indexes MS_i of already encoded transformation coefficients

5 from the input block 3105, shown in FIG. 32. On the basis of \overline{MS} , the magnitude context MC is defined as an index of an appropriate adaptive magnitude histogram $h[MC]$, which is then used for the actual encoding of the magnitude-set index MS using the range encoder in the processing block 3108. However, the local variance can enormously increase near the sharp edges, leading to a large number of histograms, and their slow adaptation. Because of that, MC is limited by the

10 constant ML , with preferable value $ML = 4$, in the processing block 3107. The number MH of magnitude histograms, i.e. the number of different MC , is preferably limited to $MH = 1 + ML = 5$. Magnitude histogram $h[MC]$ update is performed in the processing block 3109.

A slightly better prediction can be achieved in the modified processing block 3106 in a multi-pass case, by the calculation of a mean value \overline{MS} , based on all 8 neighbors from the modified

15 input block 3105.

$$\overline{MS} = \frac{1}{8} \cdot \sum_{i=0}^7 MS_i$$

It is also possible to achieve slightly better prediction in the modified processing block 3107 in a multi-pass case, based on MSP value of a "parent" transformation coefficient, i.e. one that

20 lies in the same relative spatial position, but in a higher 2DST level. MC is limited by the constant MLP , with preferable value $MLP = 3$, in the modified processing block 3107. The number MH of magnitude histograms, i.e. the number of different MC , is preferably limited to $MH = 1 + ML + MP \cdot MLP = 20$, using the constant MP , with preferable value $MP = 5$. However, the increased compression ratio obtained in this manner is negligible in relation to an

25 enormous increase in the memory size, due to the storage of all transformation coefficients.

$$MC = \min(\overline{MS}, ML) + MP \cdot \min(MSP, MLP)$$

The multi-pass method with different constant MLP is described in A. Said et al., "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Processing*, Vol. 5, No. 9, pp. 1303-1310, Sep. 1996.

In case of $MS = 0$, sign S is not encoded at all, according to the examination block 3110. Otherwise, the predictor for sign S is used to improve compression ratio. The neighborhood sign values S_i of already encoded transformation coefficients from input block 3111, shown in FIG. 33, are used for the encoding of the ternary context TC in the processing block 3112. The number of different values of ternary contexts TC is $3^4 = 81$ in a single-pass case with 4 neighborhood transformation coefficients, or $3^8 = 6561$ in a multi-pass case with 8 neighbors.

TABLE 12

| S_0 | S_1 | S_2 | S_3 | TC | SC | S_0 | S_1 | S_2 | S_3 | TC | SC | S_0 | S_1 | S_2 | S_3 | TC | SC |
|-------|-------|-------|-------|------|------|-------|-------|-------|-------|------|------|-------|-------|-------|-------|------|------|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 27 | 2 | 2 | 0 | 0 | 0 | 54 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 28 | 1 | 2 | 0 | 0 | 1 | 55 | 0 |
| 0 | 0 | 0 | 2 | 2 | 0 | 1 | 0 | 0 | 2 | 29 | 2 | 2 | 0 | 0 | 2 | 56 | 0 |
| 0 | 0 | 1 | 0 | 3 | 4 | 1 | 0 | 1 | 0 | 30 | 0 | 2 | 0 | 1 | 0 | 57 | 1 |
| 0 | 0 | 1 | 1 | 4 | 1 | 1 | 0 | 1 | 1 | 31 | 1 | 2 | 0 | 1 | 1 | 58 | 0 |
| 0 | 0 | 1 | 2 | 5 | 2 | 1 | 0 | 1 | 2 | 32 | 1 | 2 | 0 | 1 | 2 | 59 | 2 |
| 0 | 0 | 2 | 0 | 6 | 1 | 1 | 0 | 2 | 0 | 33 | 1 | 2 | 0 | 2 | 0 | 60 | 0 |
| 0 | 0 | 2 | 1 | 7 | 0 | 1 | 0 | 2 | 1 | 34 | 0 | 2 | 0 | 2 | 1 | 61 | 0 |
| 0 | 0 | 2 | 2 | 8 | 1 | 1 | 0 | 2 | 2 | 35 | 0 | 2 | 0 | 2 | 2 | 62 | 0 |
| 0 | 1 | 0 | 0 | 9 | 0 | 1 | 0 | 0 | 0 | 36 | 1 | 2 | 0 | 0 | 0 | 63 | 0 |
| 0 | 1 | 0 | 1 | 10 | 1 | 1 | 1 | 0 | 1 | 37 | 1 | 2 | 1 | 0 | 1 | 64 | 0 |
| 0 | 1 | 0 | 2 | 11 | 1 | 1 | 1 | 0 | 2 | 38 | 1 | 2 | 1 | 0 | 2 | 65 | 0 |
| 0 | 1 | 1 | 0 | 12 | 1 | 1 | 1 | 1 | 0 | 39 | 0 | 2 | 1 | 1 | 0 | 66 | 0 |
| 0 | 1 | 1 | 1 | 13 | 0 | 1 | 1 | 1 | 1 | 40 | 1 | 2 | 1 | 1 | 1 | 67 | 1 |
| 0 | 1 | 1 | 2 | 14 | 0 | 1 | 1 | 1 | 2 | 41 | 0 | 2 | 1 | 1 | 2 | 68 | 1 |
| 0 | 1 | 2 | 0 | 15 | 0 | 1 | 1 | 2 | 0 | 42 | 1 | 2 | 1 | 2 | 0 | 69 | 0 |
| 0 | 1 | 2 | 1 | 16 | 4 | 1 | 1 | 2 | 1 | 43 | 0 | 2 | 1 | 2 | 1 | 70 | 0 |
| 0 | 1 | 2 | 2 | 17 | 0 | 1 | 1 | 2 | 2 | 44 | 0 | 2 | 1 | 2 | 2 | 71 | 3 |
| 0 | 2 | 0 | 0 | 18 | 4 | 1 | 2 | 0 | 0 | 45 | 3 | 2 | 2 | 0 | 0 | 72 | 0 |
| 0 | 2 | 0 | 1 | 19 | 2 | 1 | 2 | 0 | 1 | 46 | 0 | 2 | 2 | 0 | 1 | 73 | 4 |
| 0 | 2 | 0 | 2 | 20 | 2 | 1 | 2 | 0 | 2 | 47 | 1 | 2 | 2 | 0 | 2 | 74 | 3 |
| 0 | 2 | 1 | 0 | 21 | 1 | 1 | 2 | 1 | 0 | 48 | 1 | 2 | 2 | 1 | 0 | 75 | 2 |
| 0 | 2 | 1 | 1 | 22 | 0 | 1 | 2 | 1 | 1 | 49 | 2 | 2 | 2 | 1 | 1 | 76 | 2 |
| 0 | 2 | 1 | 2 | 23 | 3 | 1 | 2 | 1 | 2 | 50 | 2 | 2 | 2 | 1 | 2 | 77 | 0 |
| 0 | 2 | 2 | 0 | 24 | 0 | 1 | 2 | 2 | 0 | 51 | 0 | 2 | 2 | 2 | 0 | 78 | 2 |
| 0 | 2 | 2 | 1 | 25 | 0 | 1 | 2 | 2 | 1 | 52 | 0 | 2 | 2 | 2 | 1 | 79 | 1 |
| 0 | 2 | 2 | 2 | 26 | 1 | 1 | 2 | 2 | 2 | 53 | 1 | 2 | 2 | 2 | 2 | 80 | 0 |

On the basis of TC , a sign context SC is defined as an index of an appropriate adaptive sign histogram $g[SC]$, which is then used for the actual encoding of sign S using the range encoder in the processing block 3116. In both cited cases, large number of different values of the sign context SC leads to histograms that would not adapt at all. Because of that, CTX table translates 81 or 6561 different values of the ternary context TC into the preferable number of 5 different values of the sign context SC per each of subbands LH, HL and HH in the processing block 3113 (TABLE 12), which simultaneously represent a number SH of sign histograms.

TABLE 13

| TC | $P(0)$ | $P(1)$ | NS | TC | $P(0)$ | $P(1)$ | NS | TC | $P(0)$ | $P(1)$ | NS |
|------|--------|--------|------|------|--------|--------|------|------|--------|--------|------|
| 0 | 0.5276 | 0.4724 | 0 | 27 | 0.6147 | 0.3853 | 0 | 54 | 0.4168 | 0.5832 | 1 |
| 1 | 0.5333 | 0.4667 | 0 | 28 | 0.4170 | 0.5830 | 1 | 55 | 0.5012 | 0.4988 | 0 |
| 2 | 0.4901 | 0.5099 | 1 | 29 | 0.6326 | 0.3674 | 0 | 56 | 0.5302 | 0.4698 | 0 |
| 3 | 0.2961 | 0.7039 | 1 | 30 | 0.4889 | 0.5111 | 1 | 57 | 0.5467 | 0.4533 | 0 |
| 4 | 0.4321 | 0.5679 | 1 | 31 | 0.4176 | 0.5824 | 1 | 58 | 0.5061 | 0.4939 | 0 |
| 5 | 0.6300 | 0.3700 | 0 | 32 | 0.4469 | 0.5531 | 1 | 59 | 0.4039 | 0.5961 | 1 |
| 6 | 0.4463 | 0.5537 | 1 | 33 | 0.5505 | 0.4495 | 0 | 60 | 0.5024 | 0.4976 | 0 |
| 7 | 0.4754 | 0.5246 | 1 | 34 | 0.5240 | 0.4760 | 0 | 61 | 0.4613 | 0.5387 | 1 |
| 8 | 0.4397 | 0.5603 | 1 | 35 | 0.4731 | 0.5269 | 1 | 62 | 0.4837 | 0.5163 | 1 |
| 9 | 0.5012 | 0.4988 | 0 | 36 | 0.4299 | 0.5701 | 1 | 63 | 0.5106 | 0.4894 | 0 |
| 10 | 0.5796 | 0.4204 | 0 | 37 | 0.5880 | 0.4120 | 0 | 64 | 0.5440 | 0.4560 | 0 |
| 11 | 0.4117 | 0.5883 | 1 | 38 | 0.5806 | 0.4194 | 0 | 65 | 0.5343 | 0.4657 | 0 |
| 12 | 0.5842 | 0.4158 | 0 | 39 | 0.4698 | 0.5302 | 1 | 66 | 0.4918 | 0.5082 | 1 |
| 13 | 0.5457 | 0.4543 | 0 | 40 | 0.4119 | 0.5881 | 1 | 67 | 0.4521 | 0.5479 | 1 |
| 14 | 0.5364 | 0.4636 | 0 | 41 | 0.5193 | 0.4807 | 0 | 68 | 0.5841 | 0.4159 | 0 |
| 15 | 0.5243 | 0.4757 | 0 | 42 | 0.4539 | 0.5461 | 1 | 69 | 0.5211 | 0.4789 | 0 |
| 16 | 0.7224 | 0.2776 | 0 | 43 | 0.4871 | 0.5129 | 1 | 70 | 0.4783 | 0.5217 | 1 |
| 17 | 0.5050 | 0.4950 | 0 | 44 | 0.4953 | 0.5047 | 1 | 71 | 0.6651 | 0.3349 | 0 |
| 18 | 0.7235 | 0.2765 | 0 | 45 | 0.3502 | 0.6498 | 1 | 72 | 0.4561 | 0.5439 | 1 |
| 19 | 0.3963 | 0.6037 | 1 | 46 | 0.4688 | 0.5312 | 1 | 73 | 0.6998 | 0.3002 | 0 |
| 20 | 0.6019 | 0.3981 | 0 | 47 | 0.5802 | 0.4198 | 0 | 74 | 0.6531 | 0.3469 | 0 |
| 21 | 0.4508 | 0.5492 | 1 | 48 | 0.4432 | 0.5568 | 1 | 75 | 0.6163 | 0.3837 | 0 |
| 22 | 0.5286 | 0.4714 | 0 | 49 | 0.3927 | 0.6073 | 1 | 76 | 0.5956 | 0.4044 | 0 |
| 23 | 0.6598 | 0.3402 | 0 | 50 | 0.6199 | 0.3801 | 0 | 77 | 0.5022 | 0.4978 | 0 |
| 24 | 0.4770 | 0.5230 | 1 | 51 | 0.5357 | 0.4643 | 0 | 78 | 0.6148 | 0.3852 | 0 |
| 25 | 0.5417 | 0.4583 | 0 | 52 | 0.4830 | 0.5170 | 1 | 79 | 0.4368 | 0.5632 | 1 |
| 26 | 0.4398 | 0.5602 | 1 | 53 | 0.4464 | 0.5536 | 1 | 80 | 0.5065 | 0.4935 | 0 |

Such incredibly small number is justified due to the encoding of a more probable sign S , which is assured by the examination block 3114, and by sign S inversion using NEG table in the processing block 3115 (TABLE 13). The ternary contexts TC with $NS = \text{NEG}[TC] = 0$ are appropriate to the probability of the positive sign $P(0)$ higher than the probability of the negative sign $P(1)$. The ternary contexts TC with $NS = \text{NEG}[TC] = 1$ are appropriate to the probability of the negative sign $P(1)$ higher than the probability of the positive sign $P(0)$. The sign histogram $g[SC]$ update is performed in the processing block 3117.

Finally, a residual R is encoded using variable length coding (VLC) in the processing block 3118 for the output in the output block 3119. The number of bits for VLC is appropriate to the coefficient range in TABLE 11, which is well known to that skilled-in-the-art.

FIG. 34 is a flowchart of the entropy decoder 29 and the decoding probability estimator 27 of this invention, based on single-pass adaptive histograms. The mean value \overline{MS} is calculated in the processing block 3402 from neighborhood magnitude-set indexes MS_i of already encoded transformation coefficients from the input block 3401, shown in FIG. 32. On the basis of \overline{MS} , the magnitude context MC is defined as an index of an appropriate adaptive magnitude histogram $h[MC]$, which is then used for the actual decoding of the magnitude-set index MS using the range decoder in the processing block 3404. MC is limited by the constant ML , with preferable value $ML = 4$, in the processing block 3403. The number MH of magnitude histograms, i.e. the number of different MC , is preferably limited to $MH = 1 + ML = 5$. Magnitude histogram $h[MC]$ update is performed in the processing block 3405.

In case of $MS = 0$, sign S is not decoded at all, according to the examination block 3406. Otherwise, neighborhood sign values S_i of already encoded transformation coefficients from input block 3407, shown in FIG. 33, are used for the decoding of a ternary context TC in the processing block 3408. On the basis of TC , a sign context SC is defined as an index of an appropriate adaptive sign histogram $g[SC]$, which is then used for the actual decoding of sign S using the range decoder in the processing block 3410. Sign histogram $g[SC]$ update is performed in the processing block 3411.

CTX table translates 81 or 6561 different values of ternary contexts TC into the preferable number of 5 different values of sign contexts SC per each of subbands LH, HL and HH in the processing block 3409 (TABLE 12), which simultaneously represent a number SH of sign

35.

histograms. Such incredibly small number is justified due to the decoding of a more probable sign S , which is assured by the examination block 3412, and by sign S inversion using NEG table in the processing block 3413 (TABLE 13).

The residual R from the input block 3414 is decoded using the decoder with variable length code (INV VLC) in the processing block 3415. The transformation coefficient value C is rebuilt using TABLE 11 in the processing block 3416 for the output in the output block 3417.

The memory size required for the storage of the context information for all subbands during the encoding or the decoding operations described in FIG. 31 and FIG. 34, is approximately three equivalent image lines, according to TABLE 14.

10

TABLE 14

| 2DST level | Context memory size per subband [coefficient] Context memory size for 3 subbands (LH, HL and HH) is 3 times higher |
|-----------------------------|---|
| 0 | $\frac{W}{2}$ |
| 1 | $\frac{W}{4}$ |
| 2 | $\frac{W}{8}$ |
| n | $\frac{W}{2^{n+1}}$ |
| $N-1$ | $\frac{W}{2^N}$ |
| All levels | $\sum_{n=0}^{N-1} \frac{W}{2^n} = W \cdot \left(1 - \frac{1}{2^N}\right) < W$ |
| All levels and all subbands | $3 \cdot \sum_{n=0}^{N-1} \frac{W}{2^n} = 3 \cdot W \cdot \left(1 - \frac{1}{2^N}\right) < 3 \cdot W$ |

The subband LL_{N-1} at the highest $N-1$ 2DST level can be directly stored without the encoding or encoded using any state-of-the-art compression method for small images. For example, LL_{N-1} can be encoded using only one context. In the first step, the first column of the transformation coefficients remains intact. In the second step, each of other transformation coefficients is replaced by the difference with its left neighbor. In the third step, each transformation coefficient from the first column, except the top-left one, is replaced by the difference with its upper neighbor. In the fourth step, the maximum MS value of all transformation coefficients is calculated, and single adaptive histogram is initialized. Finally, all

20

transformation coefficients are encoded using the same procedure defined in FIG. 31, which is used for the encoding of the transformation coefficients in all other subbands. Naturally, the decoding procedure requires the opposite order of the operations and the procedure in FIG. 34.

FIG. 35 is an initialization flowchart for histograms with fast adaptation. Each histogram bin is appropriate to a single symbol x , which can be MS for a magnitude histogram or S for a sign histogram. Simple state-of-the-art method for the probability $p(x)$ estimation of an occurrence of a symbol x is based on a number $u(x)$ of occurrences of a symbol x , and a number *Total* of occurrences of all symbols. Also, it is defined a cumulative probability $P(x)$ of all symbols y preceding symbol x in the alphabet.

$$p(x) = \frac{u(x)}{Total}$$

$$Total = \sum_x u(x)$$

$$P(x) = \sum_{y < x} p(y) = \frac{U(x)}{Total}$$

$$U(x) = \sum_{y < x} u(y).$$

The main drawback of this simple method is that *Total* is an arbitrary integer, thus requiring division operation to determine the probability $p(x)$. However, the division operation in this invention is replaced by shift right for w_3 bits, due to:

$$Total = 2^{w_3}$$

The second drawback of this simple method is a slow adaptation of the probability $p(x)$, due to averaging. The adaption of the probability $p(x)$ in this invention is provided by low-pass filtering of the binary sequence $I(j)$ representing occurrence of a symbol x in a sequence y of symbols.

$$I(j) = \begin{cases} 1, & y(j) = x \\ 0, & y(j) \neq x \end{cases}$$

The time response of the low-pass filter is extremely important for two reasons: bigger time constant provides more accurate steady-state estimation, while smaller time constant provides faster estimation. This problem is especially pronounced at the beginning of the adaptation, due to a lack of information. Instead of making a compromise in a fixed choice of a dominant pole of

the low-pass filter, the variation of a dominant pole between minimum and maximum value is implemented.

Variables are input in the input block 3501. All variables within the histogram structure h are initialized in the initialization block 3502:

5 i is the histogram bin index, with values from 1 to $imax$.

$imax$ is the maximum histogram bin index i of the non-zero histogram, in other words, $imax$ is the total number of different symbols in the alphabet, and is preferably less or equal 32 for a magnitude histogram or is equal to 2 for a sign histogram.

$h.P[]$ is a string of cumulative probabilities $h.P[i] = P(y|y < i) = \sum_{y < i} p(y)$.

10 $h.k$ is a reciprocal of an absolute dominant pole value of the low-pass filter. Its variation between $h.kmin$ and $h.kmax$ provides fast adaptation of a histogram after the start.

$h.kmax$ is a reciprocal value of a minimum absolute dominant pole of the low-pass filter, and is a fixed empirical parameter with preferable value less than $Total$.

$h.kmin$ is a reciprocal value of a maximum absolute dominant pole of the low-pass filter, and
15 is a fixed parameter with preferable value $h.kmin = 2$.

$h.i$ is the total number of symbols within the histogram plus 1.

$h.itmp$ is the temporary $h.i$ before change of $h.k$.

Step size $h.s$ is calculated in the processing block 3503, while the index i is initialized in the initialization block 3504. The histogram is initialized in the processing block 3505. The index i is
20 incremented in the processing block 3506, and examined in the examination block 3507. The last histogram bin is initialized in the processing block 3508. The same histogram initialization is used in both the encoding probability estimator 26 and the decoding probability estimator 27.

FIG. 36 is an update flowchart for histogram with fast adaptation, based on the input of the symbol x and the already described histogram structure h , in the input block 3601. However,
25 since both range encoder and range decoder cannot operate with estimated zero probability $p(x) = 0$ even for non-occurring symbols, it is necessary to modify the binary sequence $I(j)$. The modified probability $Mp(x) = Total \cdot p(x)$ is actually estimated in this invention using a fixed-point arithmetic, which is another reason for modifying the binary sequence $I(j)$. Therefore, the adaption of the probability $p(x)$ is actually performed by low-pass filtering of the modified
30 binary sequence $MI(j)$.

$$MI(j) = \begin{cases} Total - imax, & y(j) = x \\ 1, & y(j) \neq x \end{cases}$$

The maximum probability $\max p(x)$ and the minimum probability $\min p(x)$ become:

$$\max p(x) = \frac{Total - imax}{Total} < 1$$

$$\min p(x) = \frac{1}{Total} > 0$$

- 5 The preferable low-pass filter is first-order IIR filter defined as follows, where divide operation is avoided by keeping $h.k$ to be the power of two during its variation.

$$Mp(x) \leftarrow Mp(x) \cdot \left(1 - \frac{1}{h.k}\right) + MI(j)$$

- Instead of modified probability $Mp(x)$, a modified cumulative probability $MP(x) = Total \cdot P(x)$, i.e. a string of cumulative probabilities $h.P[]$ is updated. The constant K_h ,
 10 used for the fast adaptation of histograms, is initialized in the processing block 3602. The histogram bin index i is initialized in the initialization block 3603. Adding $i-1$ to the cumulative probability $h.P[i]$ prescaled with K_h in the processing block 3604 is equivalent to adding one to a number $u(x)$. The update of the cumulative probability $h.P[i]$ is performed in the processing block 3606 only for histograms with i higher than or equal to x , which probability is estimated,
 15 according to the examination block 3605. The index i is incremented in the processing block 3607, and examined in the examination block 3608.

- The second part of the histogram update algorithm is optional and especially designed for the fast adaptation of histograms. It consists of the examination block 3609, the examination block 3610, temporary save for the next update cycle in the processing block 3611, doubling of $h.k$ in
 20 the processing block 3612, increment of the total number of symbols $h.i$ within histogram h plus 1. The histogram structure h is output in the output block 3614. The mathematical equivalent of the second part of the histogram update algorithm is:

$$h.k = \min\left[2^{\lceil \log_2(h.i + h.kmin - 2) \rceil}, h.kmax\right]$$

$$h.k = \max(h.k, h.kmin),$$

- 25 where $h.kmin = 2$ preferably, which is important for the first $h.k$ during the fast adaptation.

Experimental results in FIG. 37A-37E confirmed substantial advantage of the described method for the fast adaptation of histograms, in comparison with state-of-the-art methods. Modifications of estimated probabilities are large at the beginning of the estimation process, and much smaller later, which enables the detection of small local probability variations. While the

example from FIG. 37A shows extremely large change in comparison with reality, the fast adaptation to small local probability variations provides between 10% and 20% increase in the compression ratio on standard test images.

FIG. 39 is a flowchart of the state-of-the-art range encoder, which is together with the state-of-the-art range decoder called OLD CODER, as was disclosed in G. N. N. Martin, "Range encoding: and algorithm for removing redundancy from a digitised message," *Proc. Video & Data Recording Conf.*, Southampton, UK, July 24-27, 1979; M. Schindler "A fast renormalization for arithmetic coding," *Poster at DDC, Data Compression Conf.*, Snowbird, UT, Mar. 30 – Apr. 1, 1998; and Internet location <http://www.compressconsult.com/rangecoder/>.

10 A symbol x will be encoded in the buffer of width $s = b^w$ as i :

$$\begin{aligned} i &\in \left(\lfloor s \cdot P(x) \rfloor, \lfloor s \cdot (P(x) + p(x)) \rfloor \right) \\ \lfloor s \cdot P(x) \rfloor &\leq i < \lfloor s \cdot (P(x) + p(x)) \rfloor \\ s \cdot P(x) + 1 &\leq s \cdot (P(x) + p(x)) \\ P(x) &< \frac{i+1}{s} \leq P(x) + p(x). \end{aligned}$$

15 The decoding is performed by the look-up table LUT :

$$x = LUT\left(\frac{i+1}{s}\right)$$

The coder state is represented with the following variables $(d, j), [B, B + R)$:

B = Lower range limit;

R = Range $R = T - B$, instead of T = Upper range limit;

20 d = Output byte; and

j = Number of underflow bytes.

Floating-point range encoding algorithm after the renormalization and without checking boundary conditions is: $t \leftarrow R \cdot P(x)$; $B \leftarrow B + t$; $R \leftarrow R \cdot p(x)$.

Floating-point range decoding algorithm after the renormalization and without checking boundary conditions is: $t \leftarrow B/R$; $x \leftarrow LUT(t)$; $t \leftarrow R \cdot P(x)$; $B \leftarrow B - t$; $R \leftarrow R \cdot p(x)$.

After introduction of the prescaled range r , the integer range encoding algorithm after the renormalization and without checking boundary conditions becomes:

$$r \leftarrow \left\lfloor \frac{R}{Total} \right\rfloor; t \leftarrow r \cdot U(x); B \leftarrow B + t; R \leftarrow r \cdot u(x).$$

After introducing prescaled range r , the integer range decoding algorithm after the renormalization and without checking boundary conditions becomes:

$$r \leftarrow \left\lfloor \frac{R}{Total} \right\rfloor; t \leftarrow \left\lfloor \frac{B}{r} \right\rfloor; x \leftarrow LUTr(t); t \leftarrow r \cdot U(x); B \leftarrow B - t; R \leftarrow r \cdot u(x),$$

$$\text{where } LUTr(t \cdot Total) = LUTr\left(\frac{B}{r}\right) = LUT(t).$$

- 5 The state-of-the-art range encoder and decoder algorithms are described using the arithmetic operators borrowed from C/C++ language, like:

$x \ll y$ = shift x left for y bits;

$x \gg y$ = shift x right for y bits;

$x \% y$ = remainder of x/y ;

- 10 $x|y$ = x or y ; and

$x \& y$ = x and y .

The constants *TopValue*, *BottomValue*, *ShiftBits*, *ExtraBits*, *BottomLimit* and *LowLimit* with the following preferable values are defined based on two constants w_1 and w_2 , with preferable values 8 and 32, respectively.

- 15 $TopValue = 1 \ll (w_2 - 1) = 40000000h$

$$BottomValue = TopValue \gg w_1 = 00400000h$$

$$ShiftBits = w_2 - w_1 - 1 = 23$$

$$ExtraBits = (w_2 - 2) \% w_1 + 1 = 4$$

$$BottomLimit = (1 \ll w_1) - 1 = 0FFh$$

- 20 $LowLimit = BottomLimit \ll ShiftBits$

- Basic idea with *ExtraBits* is to consider B , T and s as fixed-point values, with *ExtraBits* bits after the decimal point. This modification has been made primarily because of the decoder, where choosing *ExtraBits* in the range $(1, w_1)$ instead of more common range $(0, w_1 - 1)$ reduces the complexity of the algorithm. Otherwise, an additional examination block would be needed for testing zero value of *ExtraBits*. Before range encoding is started, the following variables must be initialized:

$$B = 0$$

$$R = TopValue$$

$$d = 0$$

$j = 0$

FIG. 38 is a flowchart of the state-of-the-art *flush* procedure, which outputs byte d in the output block 3801, initializes the loop counter i in the initialization block 3802, within the loop tests i in the examination block 3803, outputs j bytes o in the output block 3804, and increments i in the processing block 3805.

The first part of the range encoding algorithm shown in FIG. 39 performs renormalization before encoding, according to the examination block 3901. In case of a range with possible carry in the examination block 3902, first *flush* procedure 3903 defined in FIG. 38 outputs byte d and all underflow bytes $0FFh$. In case of a range with the actual carry in the examination block 3904, *flush* procedure 3905 outputs byte $d + 1$ and all underflow bytes $0h$, since $0FFh$ becomes $0h$ due to the carry. In both cases after *flush* procedures, the number of underflow bytes j is initialized in the processing block 3907 and the output byte d is generated in the processing block 3908, by assigning high byte of B . R is updated in the processing block 3909 by shifting byte which was already output or is going to be output in case of $0FFh$. B is updated in the processing block 3910 by shifting and clearing the carry. Otherwise, j is incremented in the processing block 3906.

The second part of the range encoding algorithm shown in FIG. 39 updates the range. Prescaled range r for all symbols is updated in the processing block 3911 using the first division operation. The range t of the current symbol is derived by the first multiplication operation with $U(x)$ for the current symbol x in the processing block 3912. B is updated in the processing block 3913. According to the examination block 3914, R is updated by the second multiplication operation with the $u(x)$ for the current symbol x in the processing block 3915, for all symbols except the last one. In case of the last symbol, R is updated by the subtraction operation in the processing block 3916.

Number of iterations in this loop is in the worst case $\log_b(s)$ and decreases by choosing larger b . This is the main advantage of state-of-the-art range coder over state-of-the-art arithmetic coders. For $s = 2^{31}$, an arithmetic coder uses $b = 2$, while a range coder uses $b = 2^8 = 256$. Therefore, in the worst case, an arithmetic coder needs 32 iterations per each symbol, while a range coder needs only 4 iterations. Furthermore, the first set bit of the range need not be on a fixed position. State-of-the-art binary arithmetic coders more often require renormalization of the range in comparison with the state-of-the-art range coder, thus decreasing the execution speed almost twice.

This fact is illustrated in TABLE 15 for the range coder and TABLE 16 for the arithmetic coder, which shows the encoding of the same input sequence 00001123000, with the unconditional probabilities of the symbols of $P(0)=50/100$, $P(1)=25/100$, $P(2)=15/100$ and $P(3)=10/100$. However, due to the coder delay, up to 2 more bits will be emitted latter in case of the arithmetic coder, and up to 8 more bits will be emitted later in case of the range coder.

TABLE 15

| RANGE CODER | | | | |
|-----------------|------------|------------|------------|--------------------|
| Event | Bottom | Top | Range | Out (binary) |
| Initial | 0 | 2147483648 | 2147483648 | |
| Symbol (0) | 0 | 1073741800 | 1073741800 | |
| Symbol (0) | 0 | 536870900 | 536870900 | |
| Symbol (0) | 0 | 268435450 | 268435450 | |
| Symbol (0) | 0 | 134217700 | 134217700 | |
| Symbol (1) | 67108850 | 100663275 | 33554425 | |
| Symbol (1) | 83886050 | 92274650 | 8388600 | |
| Renormalization | 2147475968 | 4294957568 | 2147481600 | 00000000 (ignored) |
| Symbol (2) | 3758087168 | 4080209408 | 322122240 | |
| Symbol (3) | 4047997148 | 4080209368 | 32212220 | |
| Symbol (0) | 4047997148 | 4064103248 | 16106100 | |
| Symbol (0) | 4047997148 | 4056050198 | 8053050 | |
| Renormalization | 1200151552 | 3261732352 | 2061580800 | 00001010 |
| Symbol (0) | 1200151552 | 2230941952 | 1030790400 | |

FIG. 40 is a flowchart of the state-of-the-art range decoder, which is together with the state-of-the-art range encoder called OLD CODER. Digits of the symbol x in base b from the input buffer are input. First $2 \cdot w_1 - \text{ExtraBits}$ bits are ignored, according to the concept of *ExtraBits*. In our case, the first byte is a dummy one. Before range decoding is started, the following variables must be initialized:

$$B = d \gg (w_1 - \text{ExtraBits})$$

$$R = 1 \ll \text{ExtraBits}$$

The first part of the range decoding algorithm shown in FIG. 40 performs renormalization before decoding, according to the examination block 4001. The appropriate bits are written into B in the processing block 4002. New symbol d is input in the input block 4003. B is updated in the processing block 4004 by shifting. R is updated in the processing block 4005 by shifting.

TABLE 16

| ARITHMETIC CODER | | | | |
|------------------|------------|------------|------------|--------------|
| Event | <i>B</i> | <i>T</i> | <i>R</i> | Out (binary) |
| Initial | 0 | 2147483648 | 2147483648 | |
| Symbol (0) | 0 | 1073741800 | 1073741800 | |
| Symbol (0) | 0 | 536870900 | 536870900 | |
| Renormalization | 0 | 1073741800 | 1073741800 | 0 (ignored) |
| Symbol (0) | 0 | 536870900 | 536870900 | |
| Renormalization | 0 | 1073741800 | 1073741800 | 0 |
| Symbol (0) | 0 | 536870900 | 536870900 | |
| Renormalization | 0 | 1073741800 | 1073741800 | 0 |
| Symbol (1) | 536870900 | 805306350 | 268435450 | |
| Renormalization | 1073741800 | 1610612700 | 536870900 | 0 |
| Renormalization | 1073741776 | 2147483576 | 1073741800 | |
| Symbol (1) | 1610612676 | 1879048126 | 268435450 | |
| Renormalization | 1073741704 | 1610612604 | 536870900 | 10 |
| Renormalization | 1073741584 | 2147483384 | 1073741800 | |
| Symbol (2) | 1879047934 | 2040109204 | 161061270 | |
| Renormalization | 1610612220 | 1932734760 | 322122540 | 10 |
| Renormalization | 1073740792 | 1717985872 | 644245080 | 1 |
| Symbol (3) | 1653561292 | 1717985792 | 64424500 | |
| Renormalization | 1159638936 | 1288487936 | 128849000 | 1 |
| Renormalization | 171794224 | 429492224 | 257698000 | 1 |
| Renormalization | 343588448 | 858984448 | 515396000 | 0 |
| Renormalization | 687176896 | 1717968896 | 1030792000 | 0 |
| Symbol (0) | 687176896 | 1202572896 | 515396000 | |
| Renormalization | 300611968 | 1331403968 | 1030792000 | |
| Symbol (0) | 300611968 | 816007968 | 515396000 | |
| Renormalization | 601223936 | 1632015936 | 1030792000 | 01 |
| Symbol (0) | 601223936 | 1116619936 | 515396000 | |

The second part of the range decoding algorithm shown in FIG. 40 updates the range. Prescaled range r for all symbols is updated in the processing block 4006 using the first division operation. The cumulative number of occurrences t of the current symbol is derived by the
5 operation. The limitation of t is performed using the examination block 4008 and the processing block 4009. After finding the symbol x based on t in the processing block 4010, t is prescaled in the processing block 4011. B is updated in the processing block 4012. According to the examination block 4013, R is updated by the second

multiplication operation with $u(x)$ for the current symbol x in the processing block 4014, for all symbols except the last one. In case of the last symbol, R is updated by the subtraction operation in the processing block 4015. After all data is decoded, a final renormalization is made.

In the state-of-the-art range encoder and the range decoder it is possible to implement first
5 division operation by $Total$ with the shift right for w_3 bits in case when $Total = 2^{w_3}$, as provided by the encoding probability estimator 26. Unfortunately, the second division operation in the processing block 4007 from FIG. 40 cannot be removed, which significantly increase the complexity of the decoding processor, since most state-of-the-art digital signal processors (DSPs) do not support division operation. Additionally, there are two multiplication operations in both
10 the encoder 30 and the decoder 31 per each symbol of the compressed image 18, which decrease the processing speed in general-purpose microprocessors. These drawbacks were eliminated in both the range encoder and the range decoder of this invention.

FIG. 41A and FIG. 41B are flowcharts of the range encoder of this invention, without division operations and optionally, without multiplication operations. The first division operation
15 by $Total = 2^{w_3}$ in the processing block 3911 in FIG. 39 is implemented by the shift right for w_3 bits in the processing block 4119 in FIG. 41B, thanks to the fast adaptation of histograms of this invention. The representation of $r = V \cdot 2^l$ according to this invention is performed in the processing block 4120 in FIG. 41B. The first multiplication operation in the processing block 3912 in FIG. 39 is implemented by the first multiplication operation with small number V and
20 shift left for l bits in the processing block 4121 in FIG. 41B. The second multiplication operation in the processing block 3915 in FIG. 39 is implemented by the second multiplication operation with small number V and shift left for l bits in the processing block 4124 in FIG. 41B. Both first and second multiplication are simplified due to less number of bits in V . Furthermore, the multiplication operation with small odd numbers $V = 3$ or $V = 5$ can be implemented by the
25 combination of shift and add operations, which is faster than a multiplication operation in a general-purpose microprocessor, thus completely removing multiplication operations. Naturally, in case of a DSP, processing block 4120 is skipped, while processing blocks 4121 and 4124 contain ordinary multiplication.

The first part of the state-of-the-art range encoding algorithm shown in FIG. 39 performing
30 renormalization before encoding, can be directly utilized in the range encoder of this invention. However, the decoupling of the processing blocks from 3901 to 3910 in FIG. 39 into two groups

of the processing blocks: first group consisting of the processing blocks from 4101 to 4110 in FIG. 41A and the second group consisting of the processing blocks from 4111 to 4118 in FIG. 41A, which actually perform processing loop, increase processing speed even further, due to the elimination of the examination block appropriate to the examination block 4104 from the second group of the processing blocks in FIG. 41A.

FIG. 42 is a flowchart of the range decoder of this invention, without division operations and optionally, without multiplication operations. The first division operation by $Total = 2^{w_3}$ in the processing block 4006 in FIG. 40 is implemented by the shift right for w_3 bits in the processing block 4206 in FIG. 42, thanks to the fast adaptation of histograms of this invention. The representation of $r = V \cdot 2^l$ according to this invention is performed in the processing block 4207 in FIG. 42, similarly to the range encoder of this invention. The first multiplication operation in the processing block 4011 in FIG. 40 is implemented by the first multiplication operation with small number V and shift left for l bits in the processing block 4212 in FIG. 42. The second multiplication operation in the processing block 4014 in FIG. 40 is implemented by the second multiplication operation with small number V and shift left for l bits in the processing block 4215 in FIG. 42. Furthermore, the multiplication operation with small odd numbers $V = 3$ or $V = 5$ can be implemented by the combination of shift and add operations, which is faster than a multiplication operation in a general-purpose microprocessor, thus completely removing multiplication operations. Naturally, in case of a DSP, processing blocks 4212 and 4215 contain ordinary multiplication.

The second division operation by r in the processing block 4007 in FIG. 40 is implemented by the division operation with small number V and shift right for l bits in the processing block 4208 in FIG. 42. The division operation by constant small odd numbers (3, 5, 9, 11, 13, 15) can be implemented with one multiplication and one right shift operations, as disclosed in D. J. Magenheimer et al., "Integer multiplication and division on the HP precision architecture," *IEEE Trans. Computers*, Vol. 37, No. 8, p. 980-990, Aug. 1988; and T. Granlud et al. "Division by invariant integers using multiplication," *SIGPLAN Notices*, Vol. 29, No. 6, p. 61, June 1994, according to TABLE 17. Specially, division operation by 7 is the most complex, since it requires the addition operation of 049240249h and the addition operation with carry and 0h (ADC with 0h), between the multiplication and the right shift operations shown in the TABLE 17.

TABLE 17

| DIVIDE BY [DECIMAL NUMBER] | MULTIPLY BY [HEXADECIMAL NUMBER] | RIGHT SHIFT FOR [BINARY DIGITS] |
|-------------------------------|-------------------------------------|------------------------------------|
| 3 | 0AAAAAAB | 1 |
| 5 | 0CCCCCCD | 2 |
| 7 | 049249249 | 1 |
| 9 | 038E38E39 | 1 |
| 11 | 0BA2E8BA3 | 3 |
| 13 | 04EC4EC4F | 2 |
| 15 | 088888889 | 3 |

It should be noticed however, that the approximations used in this invention lead to smaller compression ratio. For example, maximum possible approximation error is achieved by fixing $V = 1$ and thus completely removing all division and multiplication operations. In that case the compression ratio is decreased by less than 5%. If V is allowed to be 1 or 3, the compression ratio is decreased by less than 1%. TABLE 18 and TABLE 19 depict the difference in a number of multiplication and division operations per encoded and decoded symbol between the state-of-the-art range coder (OLD CODER) and the range coder of this invention (NEW CODER).

10

TABLE 18

| <i>Total $\neq 2^{10}$</i> | | | | | |
|---------------------------------------|-------------------|--------------|-----------------------------|--------------------|------------|
| DEVICE TYPE | OPERATION TYPE | OLD CODER | NEW CODER $r = V \cdot 2^l$ | | |
| | | | $V = 1$ | $V = 3$ $V = 5$ | $V \geq 7$ |
| ENCODER | MULTIPLY | 2 | 0 | 0 | 2 |
| | DIVIDE | 1 | 1 | 1 | 1 |
| DECODER | MULTIPLY | 2 | 0 | 1 | 3 |
| | DIVIDE | 2 | 1 | 1 | 1 |

TABLE 19

| <i>Total = 2^{10}</i> | | | | | |
|------------------------------------|-------------------|--------------|-----------------------------|--------------------|------------|
| DEVICE TYPE | OPERATION TYPE | OLD CODER | NEW CODER $r = V \cdot 2^l$ | | |
| | | | $V = 1$ | $V = 3$ $V = 5$ | $V \geq 7$ |
| ENCODER | MULTIPLY | 2 | 0 | 0 | 2 |
| | DIVIDE | 0 | 0 | 0 | 0 |
| DECODER | MULTIPLY | 2 | 0 | 1 | 3 |
| | DIVIDE | 1 | 0 | 0 | 0 |

Industrial Applicability

The communication method employing compression and decompression of this invention provides:

- Unified lossy and lossless compression;
- 5 • Still image and video intraframe (I-frame) compression;
- Symmetrical encoding & decoding time;
- Color and gray-scale image compression;
- Direct support for 4:4:4 and 4:2:2 YUV formats; and
- Integer arithmetic.

10

The novelty of this invention comprises:

- Direct and inverse non-stationary filters for the subband transformation;
- Simple context modeling for the sign and the magnitude of transformation coefficients;
- Fast adaptation of histograms for the symbol probability estimation;
- 15 • Range coder without division operations; and
- Significant reduction or the complete elimination of the synchronization buffers.

This invention eliminates the following drawbacks of state-of-the-art methods:

- Necessity for blocks, tiles or frames for low-memory performance;
- 20 • Blocky artifacts;
- Motion artifacts;
- Slow multiplication and division operations in general-purpose microprocessors;
- Absence of division operation in digital signal processors; and
- Slow execution of the arithmetic coder.

25

Typical experimental compression ratio (CR) is:

- > 3:1 lossless;
- > 100:1 visually lossless;
- > 400:1 visually lossless for large images and preprints;
- 30 • > 1000:1 slightly annoying;
- Perceived quality higher than full-frame JPEG2000; and
- PSNR higher or equal to full-frame JPEG2000.

Useful intraframe compression ratio is order of magnitude higher than in MPEG-4. Since no motion compensation is used in this invention, video (interframe) compression ratio is lower than in MPEG-4 for almost still images in video-conferencing applications. However, useful video compression ratio is higher than in MPEG-4 for medium and fast changing scenes for better perceived image quality.

This invention provides smallest encoding time (extended TABLE 20):

- 1.7..9.3 times smaller than JPEG;
- 26..152* times smaller than JPEG2000 without tiles (full frame);
- 31..183 times smaller than JPEG2000 with 128×128 tiles; and
- 252..1659 times smaller than JPEG2000 with 32×32 tiles.

This invention provides smallest decoding time (extended TABLE 21):

- 1.6..4.3 times smaller than JPEG;
- 11..105* times smaller than JPEG2000 without tiles (full frame);
- 8..82 times smaller than JPEG2000 with 128×128 tiles; and
- 57..1682 times smaller than JPEG2000 with 32×32 tiles.

This invention provides smallest encoding memory buffer (extended TABLE 22):

- 37..2357 times smaller than JPEG;
- 276..14839* times smaller than JPEG2000 without tiles (full frame);
- 94..1433 times smaller than JPEG2000 with 128×128 tiles; and
- 76..1419 times smaller than JPEG2000 with 32×32 tiles.

This invention provides smallest decoding memory buffer (extended TABLE 23):

- 32..1416 times smaller than JPEG;
- 193..13414* times smaller than JPEG2000 without tiles (full frame);
- 70..1430 times smaller than JPEG2000 with 128×128 tiles; and
- 52..1509 times smaller than JPEG2000 with 32×32 tiles.

All measurements have been made using standard set images and non-optimized C++ CIFF code (CIFF version 1.0), C++ JPEG code (The Independent JPEG Group's release 6b) and C++

All measurements have been made using standard set images and non-optimized C++ CIFF code (CIFF version 1.0), C++ JPEG code (The Independent JPEG Group's release 6b) and C++ JPEG2000 code (JasPer version 1.500.4.) with XP1800+ microprocessor and CR=20..350. The asterisk (*) denotes lack of memory in a computer with 1GB system memory.

5

TABLE 20
ENCODING TIME [s]

| IMAGE ($W \times H \times C$) | CIFF (this invention) | JPEG | JPEG2000 without tiles | JPEG2000 128×128 tiles | JPEG2000 32×32 tiles |
|------------------------------------|--------------------------|-------|---------------------------|---------------------------|-------------------------|
| 256×256×1 | 0.01 | 0.03 | 0.36 | 0.43 | 4.05 |
| 512×512×1 | 0.02 | 0.10 | 1.40 | 1.85 | 16.74 |
| 768×512×3 | 0.04 | 0.18 | 2.24 | 3.42 | 27.04 |
| 1920×1080×3 | 0.26 | 0.93 | 11.73 | 18.28 | 132.62 |
| 2048×2560×3 | 0.82 | 2.41 | 40.51 | 47.28 | 354.50 |
| 6563×3890×3 | 4.62 | 11.40 | 153.40 | 218.49 | 1605.87 |
| 10000×3543×3 | 9.08 | 15.77 | * | 282.86 | 2291.04 |

10

TABLE 21
DECODING TIME [s]

| IMAGE ($W \times H \times C$) | CIFF (this invention) | JPEG | JPEG2000 without tiles | JPEG2000 128×128 tiles | JPEG2000 32×32 tiles |
|------------------------------------|--------------------------|-------|---------------------------|---------------------------|-------------------------|
| 256×256×1 | 0.01 | 0.03 | 0.11 | 0.14 | 0.58 |
| 512×512×1 | 0.02 | 0.05 | 0.87 | 0.50 | 2.33 |
| 768×512×3 | 0.05 | 0.12 | 1.06 | 0.74 | 3.49 |
| 1920×1080×3 | 0.27 | 0.61 | 5.51 | 3.96 | 21.44 |
| 2048×2560×3 | 0.84 | 1.52 | 21.29 | 10.26 | 87.46 |
| 6563×3890×3 | 4.90 | 10.22 | 96.26 | 154.05 | 1669.44 |
| 10000×3543×3 | 9.30 | 14.47 | * | 218.60 | 3120.67 |

TABLE 22
ENCODING MEMORY BUFFER SIZE [KB]

| IMAGE ($W \times H \times C$) | CIFF (this invention) | JPEG | JPEG2000 without tiles | JPEG2000 128×128 tiles | JPEG2000 32×32 tiles |
|------------------------------------|--------------------------|--------|---------------------------|---------------------------|-------------------------|
| 256×256×1 | 8 | 376 | 3136 | 1612 | 1220 |
| 512×512×1 | 12 | 1352 | 10232 | 2196 | 1796 |
| 768×512×3 | 28 | 2412 | 15284 | 2628 | 2220 |
| 1920×1080×3 | 72 | 12364 | 78588 | 7560 | 7156 |
| 2048×2560×3 | 80 | 30940 | 194976 | 16896 | 16476 |
| 6563×3890×3 | 260 | 150620 | 858620 | 76300 | 75884 |

TABLE 23
DECODING MEMORY BUFFER SIZE [KB]

| IMAGE ($W \times H \times C$) | CIFF (this invention) | JPEG | JPEG2000 without tiles | JPEG2000 128×128 tiles | JPEG2000 32×32 tiles |
|------------------------------------|--------------------------|--------|---------------------------|---------------------------|-------------------------|
| 256×256×1 | 8 | 252 | 2268 | 1048 | 692 |
| 512×512×1 | 12 | 844 | 7764 | 1648 | 1272 |
| 768×512×3 | 28 | 2416 | 11420 | 2068 | 1668 |
| 1920×1080×3 | 72 | 12388 | 58436 | 7028 | 6936 |
| 2048×2560×3 | 80 | 30964 | 146636 | 16372 | 16848 |
| 6563×3890×3 | 260 | 150644 | 712140 | 76008 | 80180 |
| 10000×3543×3 | 380 | 208724 | * | 105144 | 111068 |

Speed measurements for encoders have been started after an input uncompressed image 10 had been read from a hard drive to the system memory, and stopped after an output compressed image 18 had been stored in the output compressed buffer 32. Speed measurements for decoders have been started after an input compressed image 19 had been read from a hard drive to the input compressed buffer 33, and stopped after an output uncompressed image 11 had been stored in the system memory. The size of system memory required for a program code, an input uncompressed image 10, an output compressed image 18, an input compressed image 19 and an output uncompressed image 11 have not been accounted in TABLE 22 and TABLE 23, leaving only memory size actually required by compression/decompression methods.

This invention implemented in hardware provides the following features:

- Lowest cost;
- Ultra-low power;
- Very low complexity;
- Absence of multipliers and dividers;
- Integer arithmetic;
- Small integrated memory; and
- Ultra-fast encoding & decoding.

Data compression has wide areas of applications, which are listed further.

Computer applications are: desktop, laptop, server, PDA, set-top box, scanner, printer, etc.

Software applications are: Web browsers, electronic commerce, desktop publishing, multimedia electronic mail, games, video help for applications, interactive imagery, etc.

Mobile phone applications are: 3G and 2.5G mobile phones, video-streaming, video-conferencing, digital camera, multimedia messaging service, single and multi-player video games, etc.

TV applications are: video-on-demand, digital TV, high-definition TV, cable TV, digital
5 video broadcast, direct satellite system, etc.

Movie applications are: digital cinema release copy, digital cinema archive, digital cinema edit master, CDROM with movie, future "DVD" format, digital video cassette, etc.

Consumer applications are: digital video camera, digital still camera, video-conferencing, videophone, color facsimile, electronic newspaper, etc.

10 Professional applications are: video surveillance, dynamic advertising, remote sensing, space imaging, interactive multimedia database, digital library and archive, etc.

Medical applications are: mammography, computerized tomography, magnetic resonance imaging, ultrasonography, X-radiography, telemedicine, etc.

Compression of medical images is typically used as an example of lossless compression.
15 However, this invention provides very high lossy compression ratio of medical images with negligible subjective and objective difference between uncompressed and decompressed image, which was experimentally confirmed by applying a specialized image processing software for a digital mammography with the same results on both uncompressed and decompressed images up to the compression ratio of 850.

20 This invention at any compression ratio provides higher or the same perceived and measured quality of the decompressed image in comparison with best state-of-the-art compression methods (JPEG2000, JPEG, MPEG-4, MPEG-2, MPEG-1 and C. Chrysafis' papers).

The fast codec embodied in software can be stored in a storage medium, such as compact disc (CD), digital versatile disc (DVD), optical disc, floppy disk, hard disk, magnetic tape, dynamic
25 random access memory (DRAM), static random access memory (SRAM), flash memory, electrically erasable programmable read only memory (EEPROM), erasable programmable read only memory (EPROM), read only memory (ROM), ferromagnetic memory, ferroelectric memory, optical storage, charge coupled devices (CCD), smart cards, etc.

Although the preferred embodiments of this invention are described and illustrated, those
30 skilled in the art can perform various modifications and design equivalents of this invention, which was already emphasized using several examples. This invention is intended to cover all such alternatives and modifications within the scope of the following claims.

CLAIMS

We claim:

- 5 1. A fast encoder for compressing input data into output compressed data, **comprising:**
at least one single-level direct subband transformer (200, 201, ...), for receiving and transforming input data to produce transformation coefficients;
at least one encoding probability estimator (260, 261, ...) coupled to appropriate said single-level direct subband transformer, for receiving the transformation coefficients and
10 estimating the probabilities of symbols within the specified contexts to produce the probabilities of symbols within the specified contexts;
at least one entropy encoder (280, 281, ...) coupled to appropriate said encoding probability estimator, for receiving and entropy encoding the transformation coefficients using the probabilities of symbols within the specified contexts to produce encoded data; and
15 an output compressed buffer (32) coupled to said entropy encoders, for receiving and substantially synchronizing the encoded data with said fast encoder to produce output compressed data,
whereby said fast encoder performs lossless compression.
- 20 2. The fast encoder of claim 1, **further comprising**
at least one quantizer (240, 241, ...) coupled to appropriate said single-level direct subband transformer, for receiving and quantizing the transformation coefficients to produce quantized transformation coefficients, **wherein:**
each said encoding probability estimator is coupled to appropriate said quantizer, for
25 receiving the quantized transformation coefficients and estimating the probabilities of symbols within the specified contexts to produce the probabilities of symbols within the specified contexts; and
each said entropy encoder is coupled to appropriate said encoding probability estimator, for receiving and entropy encoding the quantized transformation coefficients using the probabilities
30 of symbols within the specified contexts to produce encoded data,
whereby said fast encoder performs lossy compression.

3. The fast encoder of claim 1, further comprising
at least one synchronization memory (420, 421, ...) coupled to appropriate said entropy
encoder, for receiving and substantially synchronizing the encoded data with a fast encoder to
produce synchronized compressed data, wherein
- 5 said output compressed buffer is coupled to said synchronization memories, for receiving
and buffering synchronized compressed data to produce the output compressed data.
4. The fast encoder of claim 1, further comprising
at least one color space converter for converting an original input image to produce the
- 10 input data.
5. The fast encoder of claim 1, wherein:
first said single-level direct subband transformer is coupled to receive and transform the
input data to produce transformation coefficients; and
- 15 each other said single-level direct subband transformer is coupled to receive and transform
selected transformation coefficients to produce transformed transformation coefficients.
6. The fast encoder of claim 5, wherein selected transformation coefficients are low-pass
transformed for one-dimensional input data.
- 20
7. The fast encoder of claim 5, wherein selected transformation coefficients are low-pass
transformed both horizontally and vertically for two-dimensional input data.
8. The fast encoder of claim 1, wherein
- 25 said single-level direct subband transformer comprises:
at least one direct filter for horizontal filtering; and
at least one direct filter for vertical filtering.
9. The fast encoder of claim 8, wherein said direct filter for horizontal filtering is different
- 30 from said direct filter for vertical filtering.

10. The fast encoder of claim 8, wherein
at least one of said direct filter for horizontal filtering and said direct filter for vertical
filtering **comprises**
at least one direct non-stationary filter.
- 5
11. The fast encoder of claim 1, wherein
said *single-level direct subband transformer* **comprises**
at least one direct filter for filtering.
- 10 12. The fast encoder of claim 11, wherein
said direct filter **comprises**
at least one direct non-stationary filter.
13. The fast encoder of claim 12, wherein
15 said direct non-stationary filter **comprises**
a plurality of serially coupled direct non-stationary filter cells.
14. The fast encoder of claim 13, wherein
said direct non-stationary filter cell **comprises**:
20 a filter device (805);
a filter cell input x coupled to said filter device (805);
a filter cell output y coupled to said filter device (805);
a first switch (800) and a second switch (801) coupled to said filter device (805), having a
plurality of positions controlled by a clock input c ; and
25 a clock input c coupled to control said first switch (800) and said second switch (801), for
providing a non-stationarity of said direct non-stationary filter cell.
15. The fast encoder of claim 14, wherein:
said first switch (800) is in the first position for the horizontal filtering of each second pixel
30 and in the second position for the horizontal filtering of other pixels; and
said second switch (801) is in the second position for the horizontal filtering of each
second pixel and in the first position for the horizontal filtering of other pixels.

16. The fast encoder of claim 14, wherein:

said first switch (800) is in the first position for the vertical filtering of each second line and in the second position for the vertical filtering of other lines; and

5 said second switch (801) is in the second position for the vertical filtering of each second line and in the first position for the vertical filtering of other lines.

17. The fast encoder of claim 14, wherein

said direct non-stationary filter further comprises:

10 a first gain multiplier (881);

a second gain multiplier (882); and

a selection switch (880), having a plurality of positions controlled by said clock input c ,

wherein

15 an output of said plurality of serially coupled direct non-stationary filter cells is coupled to an input of said first gain multiplier (881), for multiplying said output with a first gain number to produce a first result;

an output of said plurality of serially coupled direct non-stationary filter cells is coupled to an input of said second gain multiplier (882), for multiplying said output with a second gain number to produce a second result;

20 an output of said direct non-stationary filter is coupled to an output of said first gain multiplier (881) for said selection switch (880) in the first position; and

an output of said direct non-stationary filter is coupled to an output of said second gain multiplier (882) for said selection switch (880) in the second position.

25 18. The fast encoder of claim 14, wherein said filter device comprises:

at least one delay element z^{-w} (500, 501,..., 500+m-2);

a plurality of multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1), $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2); and

a plurality of adders (700, 701, 702, 703,..., 700+m-4, 700+m-3, 700+m-2, 700+m-1),

30 wherein:

an output of each even indexed said delay element z^{-w} (500, 502,..., 500+m-4) is coupled to an input of subsequent odd indexed said delay element z^{-w} (501, 503,..., 500+m-3);

an output of each odd indexed said delay element z^{-w} (501, 503,..., 500+m-3) is coupled to an input of subsequent even indexed said delay element z^{-w} (502, 504,..., 500+m-2);

the output of each even indexed said delay element z^{-w} (500, 502,..., 500+m-2) is coupled to an input of appropriate said multiplier $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1);

5 outputs of all said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1) are coupled to inputs of said adders (701, 703,..., 700+m-3), for adding together all outputs of all said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1) to produce a first result;

inputs of first said adder (700) are coupled to receive and add the first result with said filter cell input x ;

10 an input of first said delay element z^{-w} (500) is coupled to said filter cell input x for said first switch (800) in the first position;

an input of first said delay element z^{-w} (500) is coupled to the output of first said adder (700) for said first switch (800) in the second position;

said filter cell input x and the output of each odd indexed said delay element z^{-w} (501, 15 503,..., 500+m-3) is coupled to an input of appropriate said multiplier $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2);

outputs of all said multipliers $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) are coupled to inputs of said adders (702, 704, ..., 700+m-2), for adding together all outputs of all said multipliers $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) to produce a second

20 result;

inputs of last said adder (700+m-1) are coupled to receive and add the second result with the output of last said delay element z^{-w} (500+m-2);

said filter cell output y is coupled to the output of last said delay element z^{-w} (500+m-2) for said second switch (801) in the first position; and

25 said filter cell output y is coupled to the output of last said adder (700+m-1) for said second switch (801) in the second position.

19. The fast encoder of claim 18, wherein

at least one of said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1), 30 $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) comprises

a shifting means selected from a group **consisting of:**
shifters and shifted hardwired bit line connections.

20. The fast encoder of claim 18, **wherein** said filter device **further comprises**

5 a first function N_1 means (802) coupled to receive and transform the first result to produce a third result; and

a second function N_2 means (803) coupled to receive and transform the second result to produce a fourth result,

wherein:

10 inputs of first said adder (700) are coupled to receive and add the third result with said filter cell input x ; and

inputs of last said adder (700+m-1) are coupled to receive and add the fourth result with the output of last said delay element z^{-m} (500+m-2).

15 21. The fast encoder of claim 20, **wherein**

at least one of said first function N_1 means (802) and said second function N_2 means (803) **comprises**

a shifting means selected from a group **consisting of:**
shifters and shifted hardwired bit line connections.

20

22. The fast encoder of claim 14, **wherein** said filter device **comprises:**

a delay element z^{-m} (1500);

a first multiplier (1600) and a second multiplier (1601); and

a first adder (1700) and a second adder (1701),

25 **wherein:**

an input of said first multiplier (1600) is coupled to said filter cell input x ;

an input of said second multiplier (1601) is coupled to an output of said delay element z^{-m} (1500);

30 inputs of said first adder (1700) are coupled to receive and add the output of said second multiplier (1601) with said filter cell input x ;

an input of said delay element z^{-w} (1500) is coupled to said filter cell input x for said first switch (800) in the first position;

an input of said delay element z^{-w} (1500) is coupled to the output of said first adder (1700) for said first switch (800) in the second position;

5 inputs of said second adder (1701) are coupled to receive and add the output of said first multiplier (1600) with the output of said delay element z^{-w} (1500);

said filter cell output y is coupled to the output of said delay element z^{-w} (1500) for said second switch (801) in the first position; and

said filter cell output y is coupled to the output of said second adder (1701) for said second switch (801) in the second position.

23. The fast encoder of claim 22, wherein

at least one of said first multiplier (1600) and said second multiplier (1601) comprises

a shifting means selected from a group consisting of:

15 shifters and shifted hardwired bit line connections.

24. The fast encoder of claim 23, wherein said shifting means shifts right data from its input for two bit positions to produce data at its output.

20 25. The fast encoder of claim 23, wherein said shifting means shifts right data from its input for one bit position to produce data at its output.

26. The fast encoder of claim 25, wherein

25 inputs of said first adder (1700) are coupled to receive and subtract the output of said second multiplier (1601) from said filter cell input x ; and

inputs of said second adder (1701) are coupled to receive and subtract the output of said first multiplier (1600) from the output of said delay element z^{-w} (1500).

27. The fast encoder of claim 14, wherein said filter device comprises:

30 a first delay element z^{-w} (1540), a second delay element z^{-w} (1541) and a third delay element z^{-w} (1542);

a first multiplier (1640), a second multiplier (1641), a third multiplier (1642) and a fourth multiplier (1643); and

a first adder (1740), a second adder (1741), a third adder (1742) and a fourth adder (1743),
wherein:

5 an output of said first delay element z^{-w} (1540) is coupled to an input of said second delay element z^{-w} (1541) and an input of said second multiplier (1641);

an output of said second delay element z^{-w} (1541) is coupled to an input of said third delay element z^{-w} (1542) and an input of said third multiplier (1642);

an input of said fourth multiplier (1643) is coupled to the output of said third delay element
10 z^{-w} (1542);

inputs of said second adder (1741) are coupled to receive and add the output of said second multiplier (1641) with the output of said fourth multiplier (1643);

inputs of said first adder (1740) are coupled to receive and add the output of said second adder (1741) with said filter cell input x ;

15 an input of said first delay element z^{-w} (1540) is coupled to said filter cell input x for said first switch (800) in the first position;

an input of said first delay element z^{-w} (1540) is coupled to the output of said first adder (1740) for said first switch (800) in the second position;

20 inputs of said third adder (1742) are coupled to receive and add the output of said third multiplier (1642) with the output of said first multiplier (1640);

inputs of said fourth adder (1743) are coupled to receive and add the output of said third adder (1742) with the output of said third delay element z^{-w} (1542);

said filter cell output y is coupled to the output of said third delay element z^{-w} (1542) for said second switch (801) in the first position; and

25 said filter cell output y is coupled to the output of said fourth adder (1743) for said second switch (801) in the second position.

28. The fast encoder of claim 27, wherein

at least one of said first multiplier (1640), said second multiplier (1641), said third multiplier (1642) and said fourth multiplier (1643) comprises
30

a shifting means selected from a group **consisting of:**
shifters and shifted hardwired bit line connections.

29. The fast encoder of claim 28, **wherein** said shifting means shifts right data from its input
5 for four bit positions to produce data at its output.

30. The fast encoder of claim 29, **wherein:**
inputs of said second adder (1741) are coupled to receive and subtract the output of said
second multiplier (1641) from the output of said fourth multiplier (1643); and
10 inputs of said third adder (1742) are coupled to receive and subtract the output of said third
multiplier (1642) from the output of said first multiplier (1640).

31. The fast encoder of claim 1, **wherein**
said encoding probability estimator **comprises**
15 at least one adaptive histogram updating means, for updating an adaptive histogram.

32. The fast encoder of claim 31, **wherein**
said adaptive histogram updating means **comprises**
a low-pass filter for filtering probabilities selected from a group **consisting of:**
20 probabilities of occurrences of a current symbol x ; and
cumulative probabilities of occurrences of all symbols preceding the current symbol x .

33. The fast encoder of claim 32, **wherein**
said adaptive histogram updating means **further comprises**
25 a dominant pole adapter for adapting a dominant pole of said low-pass filter.

34. The fast encoder of claim 33, **wherein**
said dominant pole adapter **comprises**
a dominant pole divider for halving a value of the dominant pole in each adaptation cycle.

30
35. The fast encoder of claim 1, **wherein**
said entropy encoder is a range encoder, **comprising**

a first multiplier for multiplying a prescaled range r with a number $Q(x)$ selected from a group consisting of:

a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a range correction $t = r \cdot U(x)$; and

5 a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.

36. The fast encoder of claim 35, wherein said first multiplier comprises:

a first simplified multiplier for multiplying a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$; and

10 a first left shifter coupled to said first simplified multiplier, for shifting left the output of said first simplified multiplier for l bit positions.

37. The fast encoder of claim 35, wherein said first multiplier comprises

a first left shifter for shifting left said number $Q(x)$ for l bit positions.

15

38. The fast encoder of claim 35, wherein said first multiplier comprises:

a third left shifter comprising:

means for zeroing its output, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1, and

means for shifting left said number $Q(x)$ for one bit position, when said small

20

number V is any odd number higher or equal 3;

a first adder coupled to said third left shifter, for adding said number $Q(x)$ with the output of said third left shifter; and

a first left shifter coupled to said first adder, for shifting left the output of said first adder for l bit positions.

25

39. The fast encoder of claim 35, wherein said first multiplier comprises:

a third left shifter comprising:

means for zeroing its output, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1,

means for shifting left said number $Q(x)$ for one bit position, when said small

30

number V is equal 3, and

means for shifting left said number $Q(x)$ for two bit positions, when said small number V is any odd number higher or equal 5;

a first adder coupled to said third left shifter, for adding said number $Q(x)$ with the output of said third left shifter; and

5 a first left shifter coupled to said first adder, for shifting left the output of said first adder for l bit positions.

40. The fast encoder of claim 1, wherein

said entropy encoder is a range encoder, comprising

10 a first divider for dividing a range R with a number *Total* of occurrences of all symbols, to produce a prescaled range $r = \lfloor R/Total \rfloor$.

41. The fast encoder of claim 40, wherein said first divider comprises

a first right shifter for shifting right said range R for $w_3 = \log_2(Total)$ bit positions.

15

42. The fast encoder of claim 1, wherein

said encoding probability estimator comprises:

a transformation coefficient C splitter into a sign S and a magnitude M ;

20 a magnitude-set index MS determinator coupled to said transformation coefficient C splitter, for determining the magnitude-set index MS using said magnitude M and a magnitude-set table;

a residual R determinator, coupled to said transformation coefficient C splitter, for determining a residual R using said magnitude M and said magnitude-set table.

25 43. The fast encoder of claim 42, wherein

said magnitude-set index MS is determined to be equal to a sum of a doubled position of the first nonzero bit of the highest significance and the value of the first next bit of the lower significance in a binary representation of said magnitude M ; and

30 said residual R is determined as the difference between said magnitude M and the lower coefficient limit, equal to a value of said magnitude M with all bits zeroed except the first nonzero bit of the highest significance and the first next bit of the lower significance in a binary representation of said magnitude M .

44. The fast encoder of claim 42, wherein
said entropy encoder comprises
a residual R encoder coupled to said residual R determinator, for encoding the residual R
5 using variable length coding (VLC).
45. The fast encoder of claim 42, wherein
said encoding probability estimator further comprises
a context modeler of a transformation coefficient to be encoded, using already encoded
10 transformation coefficients.
46. The fast encoder of claim 45, wherein
said already encoded transformation coefficients are located north-east, north, north-west
and west from said transformation coefficient to be encoded.
15
47. The fast encoder of claim 45, wherein
said encoding probability estimator further comprises
a mean value \overline{MS} determinator coupled to said context modeler, for determining \overline{MS} as
the mean value of magnitude-set indexes MS_i of said already encoded transformation
20 coefficients.
48. The fast encoder of claim 47, wherein
said encoding probability estimator further comprises
a maximum mean value \overline{MS} limiter coupled to said mean value \overline{MS} determinator, for
25 limiting a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .
49. The fast encoder of claim 48, wherein
said entropy encoder comprises
a magnitude range encoder coupled to said maximum mean value \overline{MS} limiter, for
30 encoding said magnitude-set index MS as a current symbol x , using an adaptive magnitude
histogram $h[MC]$.

50. The fast encoder of claim 48, wherein
said encoding probability estimator **further comprises**
adaptive magnitude histogram $h[MC]$ updating means coupled to said maximum mean
5 value \overline{MS} limiter, for an adaptive magnitude histogram $h[MC]$ updating using said magnitude-
set index MS as a current symbol x .
51. The fast encoder of claim 45, wherein
said encoding probability estimator **further comprises**
10 a ternary context TC determinator coupled to said transformation coefficient C splitter, for
determining a ternary context TC as the ternary code of sign values S_i of already encoded
transformation coefficients.
52. The fast encoder of claim 51, wherein
15 said encoding probability estimator **further comprises**
a sign inverter coupled to said ternary context TC determinator, for inverting less probable
said sign S using NEG table.
53. The fast encoder of claim 51, wherein
20 said encoding probability estimator **further comprises**
a ternary context TC translator coupled to said ternary context TC determinator, for
translating said ternary context TC into a sign context SC using CTX table.
54. The fast encoder of claim 53, wherein
25 said entropy encoder **comprises**
a sign range encoder coupled to said ternary context TC translator, for encoding said sign S
as a current symbol x , using an adaptive sign histogram $g[SC]$.
55. The fast encoder of claim 53, wherein
30 said encoding probability estimator **further comprises**

an adaptive sign histogram $g[SC]$ updating means coupled to said ternary context TC translator, for an adaptive sign histogram $g[SC]$ updating using said sign S as a current symbol x .

56. A fast decoder for decompressing input compressed data into output data, comprising:

5 an input compressed buffer (33), for receiving and substantially synchronizing the input compressed data with said fast decoder to produce synchronized compressed data;

10 at least one entropy decoder (290, 291, ...) coupled to said input compressed buffer and appropriate decoding probability estimator, for receiving and decoding the synchronized compressed data using the probabilities of symbols within the specified contexts to produce transformation coefficients;

 at least one decoding probability estimator (270, 271, ...) coupled to appropriate said entropy decoder, for receiving the transformation coefficients and estimating the probabilities of symbols to produce the probabilities of symbols within the specified contexts; and

15 at least one single-level inverse subband transformer (210, 211, ...) coupled to appropriate said entropy decoder, for receiving and transforming the transformation coefficients to produce the output data,

whereby said fast decoder performs lossless decompression.

57. The fast decoder of claim 56, further comprising

20 at least one dequantizer (250, 251, ...) coupled to appropriate said entropy decoder, for receiving and dequantizing the transformation coefficients to produce dequantized transformation coefficients, **wherein**

25 each said single-level inverse subband transformer is coupled to appropriate said dequantizer, for receiving and transforming the dequantized transformation coefficients to produce the output data,

whereby said fast decoder performs lossy decompression.

58. The fast decoder of claim 56, further comprising

30 at least one synchronization memory (430, 431, ...) coupled to said input compressed buffer, for receiving and substantially synchronizing buffered compressed data with said fast decoder to produce the synchronized compressed data, **wherein:**

said input compressed buffer is coupled to receive and buffer the input compressed data to produce the buffered compressed data; and

each said entropy decoder is coupled to appropriate said synchronization memory and appropriate said decoding probability estimator, for receiving and decoding the synchronized
5 compressed data using the probabilities of symbols within the specified contexts to produce transformation coefficients.

59. The fast decoder of claim 56, **further comprising**
at least one color space converter for converting the output data to produce converted
10 output data.

60. The fast decoder of claim 56, **wherein:**
last said single-level inverse subband transformer is coupled to receive and transform
transformation coefficients to produce the output data; and
15 each other said single-level inverse subband transformer is coupled to receive and transform transformation coefficients to produce selected transformation coefficients.

61. The fast decoder of claim 60, **wherein** said selected transformation coefficients are low-pass transformed for one-dimensional output data.
20

62. The fast decoder of claim 60, **wherein** said selected transformation coefficients are low-pass transformed both horizontally and vertically for two-dimensional output data.

63. The fast decoder of claim 56, **wherein**
25 said single-level inverse subband transformer **comprises:**
at least one inverse filter for horizontal filtering; and
at least one inverse filter for vertical filtering.

64. The fast decoder of claim 63, **wherein** said inverse filter for horizontal filtering is different
30 from said inverse filter for vertical filtering.

65. The fast decoder of claim 63, wherein
at least one of said inverse filter for horizontal filtering and said inverse filter for vertical
filtering **comprises**
at least one inverse non-stationary filter.
- 5 66. The fast decoder of claim 56, wherein
said single-level inverse subband transformer **comprises**
at least one inverse filter for filtering.
- 10 67. The fast decoder of claim 66, wherein
said inverse filter **comprises**
at least one inverse non-stationary filter.
- 15 68. The fast decoder of claim 67, wherein
said inverse non-stationary filter **comprises**
a plurality of serially coupled inverse non-stationary filter cells.
- 20 69. The fast decoder of claim 68, wherein
said inverse non-stationary filter cell **comprises**:
a filter device (805);
a filter cell input x coupled to said filter device (805);
a filter cell output y coupled to said filter device (805);
a first switch (800) and a second switch (801) coupled to said filter device (805), having a
plurality of positions controlled by a clock input c ; and
25 a clock input c coupled to control said first switch (800) and said second switch (801), for
providing a non-stationarity of said direct non-stationary filter cell.
- 30 70. The fast decoder of claim 69, wherein:
said first switch (800) is in the second position for the horizontal filtering of each second
pixel and in the first position for the horizontal filtering of other pixels; and
said second switch (801) is in the first position for the horizontal filtering of each second
pixel and in the second position for the horizontal filtering of other pixels.

71. The fast decoder of claim 69, wherein

said first switch (800) is in the second position for the vertical filtering of each second line and in the first position for the vertical filtering of other lines; and

5 said second switch (801) is in the first position for the vertical filtering of each second line and in the second position for the vertical filtering of other lines.

72. The fast decoder of claim 69, wherein

said direct non-stationary filter further comprises:

10 a first gain multiplier (891);

a second gain multiplier (892); and

a selection switch (890), having a plurality of positions controlled by said clock input c ,

wherein:

15 an input of said first gain multiplier (891) is coupled to an input of said inverse non-stationary filter, for multiplying an input sample with a reciprocal value of a first gain number to produce a first result;

an input of said second gain multiplier (892) is coupled to an input of said inverse non-stationary filter, for multiplying an input sample with a reciprocal value of a second gain number to produce a second result;

20 an input of said plurality of serially coupled inverse non-stationary filter cells is coupled to an output of said first gain multiplier (891), for said selection switch (890) in the second position; and

an input of said plurality of serially coupled inverse non-stationary filter cells is coupled to an output of said second gain multiplier (892), for said selection switch (890) in the first position.

25

73. The fast decoder of claim 69, wherein said filter device comprises:

at least one delay element z^{-w} (500, 501,..., 500+m-2);

a plurality of multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1), $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2); and

30 a plurality of adders (700, 701, 702, 703,..., 700+m-4, 700+m-3, 700+m-2, 700+m-1),

wherein:

an output of each even indexed said delay element z^{-w} (500, 502,..., 500+m-4) is coupled to an input of subsequent odd indexed said delay element z^{-w} (501, 503,..., 500+m-3);

an output of each odd indexed said delay element z^{-w} (501, 503,..., 500+m-3) is coupled to an input of subsequent even indexed said delay element z^{-w} (502, 504,..., 500+m-2);

the output of each even indexed said delay element z^{-w} (500, 502,..., 500+m-2) is coupled to an input of appropriate said multiplier $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1);

outputs of all said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1) are coupled to inputs of said adders (701, 703,..., 700+m-3), for adding together all outputs of all said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1) to produce a first result;

inputs of first said adder (700) are coupled to receive and add the first result with said filter cell input x ;

an input of first said delay element z^{-w} (500) is coupled to said filter cell input x for said first switch (800) in the first position;

an input of first said delay element z^{-w} (500) is coupled to the output of first said adder (700) for said first switch (800) in the second position;

said filter cell input x and the output of each odd indexed said delay element z^{-w} (501, 503,..., 500+m-3) is coupled to an input of appropriate said multiplier $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2);

outputs of all said multipliers $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) are coupled to inputs of said adders (702, 704, ..., 700+m-2), for adding together all outputs of all said multipliers $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) to produce a second result;

inputs of last said adder (700+m-1) are coupled to receive and add the second result with the output of last said delay element z^{-w} (500+m-2);

said filter cell output y is coupled to the output of last said delay element z^{-w} (500+m-2) for said second switch (801) in the first position; and

said filter cell output y is coupled to the output of last said adder (700+m-1) for said second switch (801) in the second position.

74. The fast decoder of claim 73, wherein

at least one of said multipliers $K_1[0]$ (601), $K_1[1]$ (603),..., $K_1[k-1]$ (600+m-1),
 $K_2[k-1]$ (600), $K_2[k-2]$ (602),..., $K_2[0]$ (600+m-2) comprises

a shifting means selected from a group consisting of:

5 shifters and shifted hardwired bit line connections.

75. The fast decoder of claim 73, wherein

said inverse non-stationary filter cell **further comprises**

a first function N_1 means (802) coupled to receive and transform the first result to produce
10 a third result; and

a second function N_2 means (803) coupled to receive and transform the second result to
produce a fourth result,

wherein:

inputs of first said adder (700) are coupled to receive and add the third result with said
15 filter cell input x ; and

inputs of last said adder (700+m-1) are coupled to receive and add the fourth result with
the output of last said delay element z^{-m} (500+m-2).

76. The fast decoder of claim 75, wherein

20 at least one of said first function N_1 means (802) and said second function N_2 means
(803) comprises

a shifting means selected from a group consisting of:

shifters and shifted hardwired bit line connections.

25 77. The fast decoder of claim 69, wherein said filter device comprises:

a delay element z^{-m} (1510);

a first multiplier (1610) and a second multiplier (1611); and

a first adder (1710) and a second adder (1711),

wherein:

30 an input of said first multiplier (1610) is coupled to said filter cell input x ;

an input of said second multiplier (1611) is coupled to an output of said delay element z^{-w} (1510);

inputs of said first adder (1710) are coupled to receive and add the output of said second multiplier (1611) with said filter cell input x ;

5 an input of said delay element z^{-w} (1510) is coupled to said filter cell input x , for said first switch (800) in the first position;

an input of said delay element z^{-w} (1510) is coupled to the output of said first adder (1710), for said first switch (800) in the second position;

10 inputs of said second adder (1711) are coupled to receive and add the output of said first multiplier (1610) with the output of said delay element z^{-w} (1510);

said filter cell output y is coupled to the output of said delay element z^{-w} (1510), for said second switch (801) in the first position; and

said filter cell output y is coupled to the output of said second adder (1711), for said second switch (801) in the second position.

15

78. The fast decoder of claim 77, wherein

at least one of said first multiplier (1610) and said second multiplier (1611) comprises a shifting means selected from a group consisting of:
shifters and shifted hardwired bit line connections.

20

79. The fast decoder of claim 78, wherein said shifting means shifts right data from its input for one bit position to produce data at its output.

80. The fast decoder of claim 78, wherein said shifting means shifts right data from its input for two bit positions to produce data at its output.

25

81. The fast decoder of claim 80, wherein

inputs of said first adder (1710) are coupled to receive and subtract the output of said second multiplier (1611) from said filter cell input x ; and

30 inputs of said second adder (1711) are coupled to receive and subtract the output of said first multiplier (1610) from the output of said delay element z^{-w} (1510).

82. The fast decoder of claim 69, wherein said filter device comprises:

a first delay element z^{-w} (1550), a second delay element z^{-w} (1551) and a third delay element z^{-w} (1552);

5 a first multiplier (1650), a second multiplier (1651), a third multiplier (1652) and a fourth multiplier (1653); and

a first adder (1750), a second adder (1751), a third adder (1752) and a fourth adder (1753),
wherein:

an output of said first delay element z^{-w} (1550) is coupled to an input of said second delay element z^{-w} (1551) and an input of said second multiplier (1651);

an output of said second delay element z^{-w} (1551) is coupled to an input of said third delay element z^{-w} (1552) and an input of said third multiplier (1652);

an input of said fourth multiplier (1653) is coupled to the output of said third delay element z^{-w} (1552);

15 inputs of said second adder (1751) are coupled to receive and add the output of said fourth multiplier (1653) with the output of said second multiplier (1651);

inputs of said first adder (1750) are coupled to receive and add the output of said second adder (1751) with said filter cell input x ;

an input of said first delay element z^{-w} (1550) is coupled to said filter cell input x , for said
20 first switch (800) in the first position;

an input of said first delay element z^{-w} (1550) is coupled to the output of said first adder (1750), for said first switch (800) in the second position;

inputs of said third adder (1752) are coupled to receive and add the output of said first multiplier (1650) with the output of said third multiplier (1652);

25 inputs of said fourth adder (1753) are coupled to receive and add the output of said third adder (1752) with the output of said third delay element z^{-w} (1552);

said filter cell output y is coupled to the output of said third delay element z^{-w} (1552), for said second switch (801) in the first position; and

said filter cell output y is coupled to the output of said fourth adder (1753), for said second
30 switch (801) in the second position.

83. The fast decoder of claim 82, wherein
at least one of said first multiplier (1650), said second multiplier (1651), said third
multiplier (1652) and said fourth multiplier (1653) comprises
a shifting means selected from a group consisting of:
5 shifters and shifted hardwired bit line connections.
84. The fast decoder of claim 83, wherein said shifting means shifts right data from its input
for four bit positions to produce data at its output.
- 10 85. The fast decoder of claim 84, wherein:
inputs of said second adder (1751) are coupled to receive and subtract the output of said
fourth multiplier (1653) from the output of said second multiplier (1651); and
inputs of said third adder (1752) are coupled to receive and subtract the output of said first
multiplier (1650) from the output of said third multiplier (1652).
15
86. The fast decoder of claim 56, wherein
said decoding probability estimator comprises
at least one adaptive histogram updating means, for updating an adaptive histogram.
- 20 87. The fast decoder of claim 86, wherein
said adaptive histogram updating means comprises
a low-pass filter for filtering probabilities selected from a group consisting of:
probabilities of occurrences of a current symbol x ; and
cumulative probabilities of occurrences of all symbols preceding the current symbol x .
25
88. The fast decoder of claim 87, wherein
said adaptive histogram updating means further comprises
a dominant pole adapter for adapting a dominant pole of said low-pass filter.
- 30 89. The fast decoder of claim 88, wherein
said dominant pole adapter comprises
a dominant pole divider for halving a value of the dominant pole in each adaptation cycle.

90. The fast decoder of claim 56, wherein
said entropy decoder is a range decoder, comprising
a first multiplier for multiplying a prescaled range r with a number $Q(x)$ selected from a
5 group consisting of:
a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.
- 10 91. The fast decoder of claim 90, wherein said first multiplier comprises:
a first simplified multiplier for multiplying a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number
 $Q(x)$; and
a first left shifter coupled to said first simplified multiplier, for shifting left the output of
said first simplified multiplier for l bit positions.
- 15 92. The fast decoder of claim 90, wherein said first multiplier comprises
a first left shifter for shifting left said number $Q(x)$ for l bit positions.
93. The fast decoder of claim 90, wherein said first multiplier comprises:
20 a third left shifter comprising:
means for zeroing its output, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1, and
means for shifting left said number $Q(x)$ for one bit position, when said small
number V is any odd number higher or equal 3;
a first adder coupled to said third left shifter, for adding said number $Q(x)$ with the output
25 of said third left shifter; and
a first left shifter coupled to said first adder, for shifting left the output of said first adder
for l bit positions.
94. The fast decoder of claim 90, wherein said first multiplier comprises:
30 a third left shifter comprising:

means for zeroing its output, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1,

means for shifting left said number $Q(x)$ for one bit position, when said small number V is equal 3, and

means for shifting left said number $Q(x)$ for two bit positions, when said small

5 number V is any odd number higher or equal 5;

a first adder coupled to said third left shifter, for adding said number $Q(x)$ with the output of said third left shifter; and

a first left shifter coupled to said first adder, for shifting left the output of said first adder for l bit positions.

10

95. The fast decoder of claim 56, wherein

said entropy decoder is a range decoder, comprising

a first divider for dividing a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled range $r = \lfloor R/Total \rfloor$.

15

96. The fast decoder of claim 95, wherein said first divider comprises

a first right shifter for shifting right said range R for $w_3 = \log_2(Total)$ bit positions.

97. The fast decoder of claim 56, wherein

20 said entropy decoder is a range decoder, comprising

a second divider for dividing a bottom range limit B with a prescaled range r , to produce a range correction $t = \lfloor B/r \rfloor$.

98. The fast decoder of claim 97, wherein said second divider comprises:

25 a second simplified divider for dividing said bottom range limit B with a small number

$V = \lfloor r \cdot 2^{-l} \rfloor$; and

a second right shifter coupled to said second simplified divider, for shifting right the output of said second simplified divider for l bit positions.

99. The fast decoder of claim 97, wherein said second divider comprises:

a third multiplier for multiplying said bottom range limit B with a first predefined number, dependent on a small number $V = \lfloor r \cdot 2^{-l} \rfloor$; and

5 a second right shifter coupled to said third multiplier, for shifting right the output of said third multiplier for a sum of l and a second predefined number of bit positions, dependent on said small number V .

100. The fast decoder of claim 56, wherein

said decoding probability estimator comprises

10 a transformation coefficient C builder for reconstructing transformation coefficient C using a magnitude-set index MS , a sign S and a residual R .

101. The fast decoder of claim 100, wherein

said entropy decoder comprises

15 a residual R decoder, for decoding said residual R using inverse variable length coding (INV VLC).

102. The fast decoder of claim 100, wherein

said decoding probability estimator further comprises

20 a context modeler of a transformation coefficient to be decoded, using already decoded transformation coefficients.

103. The fast decoder of claim 102, wherein

25 said already decoded transformation coefficients are located north-east, north, north-west and west from said transformation coefficient to be decoded.

104. The fast decoder of claim 102, wherein

said decoding probability estimator further comprises

30 a mean value \overline{MS} determinator coupled to said context modeler, for determining \overline{MS} as the mean value of magnitude-set indexes MS_i of said already decoded transformation coefficients.

105. The fast decoder of claim 104, wherein

said decoding probability estimator further comprises

a maximum mean value \overline{MS} limiter coupled to said mean value \overline{MS} determinator, for limiting a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .

5

106. The fast decoder of claim 105, wherein

said entropy decoder comprises

a magnitude range decoder coupled to said maximum mean value \overline{MS} limiter, for decoding a magnitude-set index MS as a current symbol x , using an adaptive magnitude

10 histogram $h[MC]$.

107. The fast decoder of claim 106, wherein

said decoding probability estimator further comprises

adaptive magnitude histogram $h[MC]$ updating means coupled to said magnitude range
15 decoder, for an adaptive magnitude histogram $h[MC]$ updating using decoded said magnitude-set index MS as a current symbol x .

108. The fast decoder of claim 102, wherein

said decoding probability estimator further comprises

20 a ternary context TC determinator coupled to said context modeler, for determining a ternary context TC as the ternary code of sign values S_i of already decoded transformation coefficients.

109. The fast decoder of claim 108, wherein

25 said decoding probability estimator further comprises

a ternary context TC translator coupled to said ternary context TC determinator, for translating ternary context TC into a sign context SC using CTX table.

110. The fast decoder of claim 109, wherein

30 said entropy decoder comprises

a sign range decoder coupled to said ternary context TC translator, for decoding a sign S as a current symbol x , using an adaptive sign histogram $g[SC]$.

111. The fast decoder of claim 110, wherein

5 said decoding probability estimator **further comprises**
 adaptive sign histogram $g[SC]$ updating means coupled to said sign range decoder, for an
adaptive sign histogram $g[SC]$ updating using decoded said sign S as a current symbol x .

112. The fast decoder of claim 110, wherein

10 said decoding probability estimator **further comprises**
 a sign inverter coupled to said sign range decoder, for inverting less probable said sign S
using NEG table.

113. A method for fast encoding input data into output compressed data, **comprising:**

15 direct subband transforming the input data to produce transformation coefficients;
 estimating the probabilities of symbols within the specified contexts using the
transformation coefficients to produce the probabilities of symbols within the specified contexts;
 entropy encoding transformation coefficients using the probabilities of symbols within the
specified contexts to produce encoded data; and
20 substantially synchronizing the encoded data to produce output compressed data,
whereby said method for fast encoding performs lossless compression.

114. The method for fast encoding of claim 113, **further comprising**

 quantizing transformation coefficients to produce quantized transformation coefficients,
25 **wherein:**
 said estimating the probabilities of symbols within the specified contexts is performed
using the quantized transformation coefficients to produce the probabilities of symbols within the
specified contexts; and
 said entropy encoding quantization transformation coefficients is performed using the
30 probabilities of symbols within the specified contexts to produce encoded data,
whereby said method for fast encoding performs lossy compression.

115. The method for fast encoding of claim 113, **further comprising**
substantially synchronizing the encoded data to produce synchronized compressed data,
wherein
buffering the synchronized compressed data is performed to produce output compressed
5 data.

116. The method for fast encoding of claim 113, **further comprising**
color space converting original input data to produce input data.

10 117. The method for fast encoding of claim 113, **wherein**
said direct subband transforming **comprises**:
(a) direct subband transforming the input data to produce transformation coefficients;
(b) direct subband transforming selected transformation coefficients to produce
transformed transformation coefficients; and
15 (c) repeating step (b) finite number of times.

118. The method for fast encoding of claim 117, **wherein** said selected transformation
coefficients are low-pass transformed for one-dimensional input data.

20 119. The method for fast encoding of claim 117, **wherein** said selected transformation
coefficients are low-pass transformed both horizontally and vertically for two-dimensional input
data.

25 120. The method for fast encoding of claim 113, **wherein**
said direct subband transforming **comprises**:
at least one horizontal direct filtering; and
at least one vertical direct filtering.

30 121. The method for fast encoding of claim 120, **wherein** said horizontal direct filtering is
different from said vertical direct filtering.

122. The method for fast encoding of claim 120, wherein
at least one of said horizontal direct filtering and said vertical direct filtering comprises
at least one direct non-stationary filtering.
- 5 123. The method for fast encoding of claim 113, wherein
said direct subband transforming comprises
at least one direct filtering.
- 10 124. The method for fast encoding of claim 123, wherein
said direct filtering comprises
at least one direct non-stationary filtering.
- 15 125. The method for fast encoding of claim 124, wherein
said direct non-stationary filtering comprises
a plurality of successive direct non-stationary cell filtering steps.
- 20 126. The method for fast encoding of claim 125, wherein
said direct non-stationary cell filtering comprises:
filtering using first direct transfer function in the first cycle; and
filtering using second direct transfer function in the second cycle.
- 25 127. The method for fast encoding of claim 126, wherein
said first cycle is active during horizontal filtering of each second pixel; and
said second cycle is active during horizontal filtering of other pixels.
128. The method for fast encoding of claim 126, wherein
said first cycle is active during vertical filtering of each second line; and
said second cycle is active during vertical filtering of other lines.
- 30 129. The method for fast encoding of claim 126, further comprising:
first multiplying a result of said plurality of successive direct non-stationary cell filtering
steps with a first gain number to produce a first result;

second multiplying a result of said plurality of successive direct non-stationary cell filtering steps with a second gain number to produce a second result;

selecting the first result in each first cycle to produce an output sample; and

selecting the second result in each second cycle to produce the output sample.

5

130. The method for fast encoding of claim 126, wherein

said direct non-stationary cell filtering **further comprises:**

delaying an input sample for w samples to produce a plurality of even and odd indexed delayed results in each first cycle;

10 multiplying each even indexed delayed result with appropriate first filter coefficient selected from a group of first filter coefficients to produce first results;

adding together all first results to produce a third result;

adding the third result with the input sample to produce a fifth result;

15 delaying the fifth result for w samples to produce a plurality of even and odd indexed delayed results in each second cycle;

multiplying the input sample and each odd indexed delayed result with appropriate second filter coefficient selected from a group of second filter coefficients to produce second results;

adding together all second results to produce a fourth result;

adding the fourth result with last delayed result to produce a sixth result;

20 outputting the sixth result in each first cycle; and

outputting the last delayed result in each second cycle.

131. The method for fast encoding of claim 130, wherein

at least one said multiplying **comprises**

25 an operation selected from a group **consisting of:**

shifting and bit remapping.

132. The method for fast encoding of claim 130, wherein

said direct non-stationary cell filtering **further comprises:**

30 transforming the third result by first function N_1 to produce a seventh result;

transforming the fourth result by first function N_2 to produce an eight result;

adding the seventh result with the input sample to produce a fifth result; and
adding the eight result with last delayed result to produce a sixth result.

133. The method for fast encoding of claim 132, wherein
5 at least one said transforming comprises
an operation selected from a group consisting of:
shifting and bit remapping.

134. The method for fast encoding of claim 126, wherein
10 said direct non-stationary cell filtering further comprises:
delaying an input sample for w samples to produce a delayed result in each first cycle;
second multiplying the delayed result with a second filter coefficient to produce a second
result;
first adding the second result with the input sample to produce a fourth result;
15 delaying the fourth result for w samples to produce the delayed result in each second cycle;
first multiplying the input sample with a first filter coefficient to produce a first result;
second adding the first result with the delayed result to produce a third result;
outputting the third result in each first cycle; and
outputting the delayed result in each second cycle.

20

135. The method for fast encoding of claim 134, wherein
at least one of said first multiplying and said second multiplying comprises
an operation selected from a group consisting of:
shifting and bit remapping.

25

136. The method for fast encoding of claim 135, wherein said operation comprises
shifting right for two bit positions.

137. The method for fast encoding of claim 135, wherein said operation comprises
30 shifting right for one bit position.

138. The method for fast encoding of claim 137, wherein

said first adding **comprises** subtracting the second result from the input sample to produce a fourth result; and

5 said second adding **comprises** subtracting the first result from the delayed result to produce a third result.

139. The method for fast encoding of claim 126, wherein

said direct non-stationary cell filtering **further comprises**:

10 delaying the input sample for w samples to produce a first delayed result in each first cycle;

 delaying the first delayed result for w samples to produce a second delayed result;

 delaying the second delayed result for w samples to produce a third delayed result;

 first multiplying the input sample with a first filter coefficient to produce a first result;

 second multiplying the first delayed result with a second filter coefficient to produce a second result;

15 third multiplying the second delayed result with a third filter coefficient to produce a third result;

 fourth multiplying the third delayed result with a fourth filter coefficient to produce a fourth result;

 second adding the second result with the fourth result to produce a sixth result;

20 third adding the third result with the first result to produce a fifth result;

 first adding the sixth result with the input sample to produce an eight result;

 delaying the eight result for w samples to produce the first delayed result in each second cycle;

 fourth adding the fifth result with the third delayed result to produce a seventh result;

25 outputting the seventh result in each first cycle; and

 outputting the third delayed result in each second cycle.

140. The method for fast encoding of claim 139, wherein

at least one of said first multiplying, said second multiplying, said third multiplying and

30 said fourth multiplying **comprises**

an operation selected from a group consisting of:

shifting and bit remapping.

141. The method for fast encoding of claim 140, wherein said operation comprises shifting right for four bit positions.

5 142. The method for fast encoding of claim 141, wherein:

said second adding comprises subtracting the second result from the fourth result to produce a sixth result; and

said third adding comprises subtracting the third result from the first result to produce a fifth result.

10

143. The method for fast encoding of claim 113, wherein

said estimating the probabilities of symbols within the specified contexts comprises updating adaptive histograms.

15 144. The method for fast encoding of claim 143, wherein

said updating adaptive histograms comprises

low-pass filtering probabilities selected from a group consisting of:

probabilities of occurrences of a current symbol x ; and

cumulative probabilities of occurrences of all symbols preceding the current symbol x .

20

145. The method for fast encoding of claim 144, wherein

said updating adaptive histograms further comprises

adapting a dominant pole during said low-pass filtering.

25 146. The method for fast encoding of claim 145, wherein

said adapting a dominant pole comprises

halving value of the dominant pole in each adaptation cycle.

147. The method for fast encoding of claim 113, wherein

30 said entropy encoding is range encoding, comprising

first multiplying a prescaled range r with a number $Q(x)$ selected from a group consisting of:

a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a range correction $t = r \cdot U(x)$; and

a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.

5 **148.** The method for fast encoding of claim 147, wherein said first multiplying comprises:

 simplified multiplying a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a first result; and

 shifting left the first result for l bit positions.

10 **149.** The method for fast encoding of claim 147, wherein said first multiplying comprises

 shifting left said number $Q(x)$ for l bit positions.

150. The method for fast encoding of claim 147, wherein said first multiplying comprises:

 zeroing a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;

15 shifting left said number $Q(x)$ for one bit position to produce a first result, when said small number V is any odd number higher or equal 3;

 adding said number $Q(x)$ with the first result to produce a second result; and

 shifting left the second result for l bit positions.

20 **151.** The method for fast encoding of claim 147, wherein said first multiplying comprises:

 zeroing a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;

 shifting left said number $Q(x)$ for one bit position to produce a first result, when said small number V is equal 3;

 shifting left said number $Q(x)$ for two bit positions to produce a first result, when said small number V is any odd number higher or equal 5;

25 adding said number $Q(x)$ with the first result to produce a second result; and

 shifting left the second result for l bit positions.

152. The method for fast encoding of claim 113, wherein

30 said entropy encoding is range encoding, comprising

first dividing a range R with a number *Total* of occurrences of all symbols, to produce a prescaled range $r = \lfloor R/Total \rfloor$.

153. The method for fast encoding of claim 152, wherein said first dividing comprises
5 shifting right said range R for $w_3 = \log_2(Total)$ bit positions.
154. The method for fast encoding of claim 113, wherein
said estimating the probabilities of symbols within the specified contexts comprises
splitting a transformation coefficient C into a sign S and a magnitude M ;
10 determining a magnitude-set index MS using said magnitude M and a magnitude-set table;
and
determining a residual R using said magnitude M and said magnitude-set table.
155. The method for fast encoding of claim 154, wherein
15 said magnitude-set index MS is determined to be equal to a sum of a doubled position of
the first nonzero bit of the highest significance and the value of the first next bit of the lower
significance in a binary representation of said magnitude M ; and
said residual R is determined as the difference between said magnitude M and the lower
coefficient limit, equal to a value of said magnitude M with all bits zeroed except the first
20 nonzero bit of the highest significance and the first next bit of the lower significance in a binary
representation of said magnitude M .
156. The method for fast encoding of claim 154, wherein
said entropy encoding comprises
25 encoding a residual R using variable length coding (VLC).
157. The method for fast encoding of claim 154, wherein
said estimating the probabilities of symbols within the specified contexts further
comprises
30 context modeling a transformation coefficient to be encoded, using already encoded
transformation coefficients.

158. The method for fast encoding of claim 157, wherein

said already encoded transformation coefficients are located north-east, north, north-west and west from said transformation coefficient to be encoded.

5 159. The method for fast encoding of claim 157, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

determining a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said already encoded transformation coefficients.

10

160. The method for fast encoding of claim 159, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

limiting a maximum mean value \overline{MS} by a constant ML to produce a magnitude context
15 MC .

161. The method for fast encoding of claim 160, wherein

said entropy encoding **comprises**

range encoding said magnitude-set index MS as a current symbol x , using an adaptive
20 magnitude histogram $h[MC]$.

162. The method for fast encoding of claim 160, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

25 updating of an adaptive magnitude histogram $h[MC]$ using said magnitude-set index MS as a current symbol x .

163. The method for fast encoding of claim 162, wherein

30 said estimating the probabilities of symbols within the specified contexts **further comprises**

determining a ternary context TC as the ternary code of sign values S_i of said already encoded transformation coefficients.

164. The method for fast encoding of claim 163, wherein

5 said estimating the probabilities of symbols within the specified contexts **further comprises**
 inverting less probable said sign S using NEG table.

165. The method for fast encoding of claim 163, wherein

10 said estimating the probabilities of symbols within the specified contexts **further comprises**
 translating said ternary context TC into a sign context SC using CTX table.

166. The method for fast encoding of claim 165, wherein

15 said entropy encoder **comprises**
 range encoding said sign S as a current symbol x , using an adaptive sign histogram $g[SC]$.

167. The method for fast encoding of claim 165, wherein

20 said estimating the probabilities of symbols within the specified contexts **further comprises**
 updating of an adaptive sign histogram $g[SC]$ using said sign S as a current symbol x .

168. A method for fast decoding of input compressed data into output data, comprising:

25 substantially synchronizing the input compressed data to produce synchronized compressed data;

 entropy decoding the synchronized compressed data using the probabilities of symbols within the specified contexts to produce transformation coefficients;

 estimating the probabilities of symbols within the specified contexts using the transformation coefficients to produce the probabilities of symbols within the specified contexts;

30 and

 inverse subband transforming the transformation coefficients to produce the output data,
whereby said method for fast decoding performs lossless decompression.

169. The method for fast decoding of claim 168, **further comprising**
dequantizing transformation coefficients to produce dequantized transformation
coefficients, **wherein**
5 said inverse subband transforming the dequantized transformation coefficients is performed
to produce the output data,
whereby said method for fast decoding performs lossy decompression.
170. The method for fast decoding of claim 168, **further comprising**
10 buffering the input compressed data to produce buffered compressed data, **wherein**
 said substantially synchronizing the buffered compressed data is performed to produce
synchronized compressed data.
171. The method for fast decoding of claim 168, **further comprising**
15 color space converting the output data to produce converted output data.
172. The method for fast decoding of claim 168, **wherein**
 said inverse subband transforming **comprises**:
 (a) inverse subband transforming transformation coefficients to produce selected
20 transformation coefficients;
 (b) repeating step (a) finite number of times; and
 (c) inverse subband transforming transformation coefficients to produce the output data.
173. The method for fast decoding of claim 172, **wherein** said selected transformation
25 coefficients are low-pass transformed for one-dimensional output data.
174. The method for fast decoding of claim 172, **wherein** said selected transformation
coefficients are low-pass transformed both horizontally and vertically for two-dimensional output
data.
30
175. The method for fast decoding of claim 168, **wherein**
 said inverse subband transforming **comprises**:

at least one horizontal inverse filtering; and
at least one vertical inverse filtering.

176. The method for fast decoding of claim 175, wherein said horizontal inverse filtering is
5 different from said vertical inverse filtering.

177. The method for fast decoding of claim 175, wherein
at least one of said horizontal inverse filtering and said vertical inverse filtering comprises
at least one inverse non-stationary filtering.

10

178. The method for fast decoding of claim 168, wherein
said inverse subband transforming comprises
at least one inverse filtering.

15 179. The method for fast decoding of claim 178, wherein
said inverse filtering comprises
at least one inverse non-stationary filtering.

180. The method for fast decoding of claim 179, wherein
20 said inverse non-stationary filtering comprises
a plurality of successive inverse non-stationary cell filtering steps.

181. The method for fast decoding of claim 180, wherein
said inverse non-stationary cell filtering comprises:
25 filtering using first inverse transfer function in the first cycle; and
filtering using second inverse transfer function in the second cycle.

182. The method for fast decoding of claim 181, wherein
said second cycle is active during horizontal filtering of each second pixel; and
30 said first cycle is active during horizontal filtering of other pixels.

183. The method for fast decoding of claim 181, wherein
said second cycle is active during vertical filtering of each second line; and
said first cycle is active during vertical filtering of other lines.
- 5 184. The method for fast decoding of claim 181, wherein
said inverse non-stationary cell filtering **further comprising:**
first multiplying an input with a reciprocal value of a first gain number to produce a first
result;
second multiplying an input with a reciprocal value of the second gain number to produce a
10 second result;
selecting first result in each second cycle to produce an input sample for said plurality of
successive inverse non-stationary cell filtering steps; and
selecting second result in each first cycle to produce the input sample for said plurality of
successive inverse non-stationary cell filtering steps.
- 15 185. The method for fast decoding of claim 181, wherein
said inverse non-stationary cell filtering **further comprises:**
delaying an input sample for w samples to produce a plurality of even and odd indexed
delayed results in each first cycle;
20 multiplying each even indexed delayed result with appropriate first filter coefficient
selected from a group of first filter coefficients to produce first results;
adding together all first results to produce a third result;
adding the third result with the input sample to produce a fifth result;
delaying the fifth result for w samples to produce a plurality of even and odd indexed
25 delayed results in each second cycle;
multiplying the input sample and each odd indexed delayed result with appropriate second
filter coefficient selected from the group of second filter coefficients to produce second results;
adding together all second results to produce a fourth result;
adding the fourth result with last delayed result to produce a sixth result;
30 outputting the sixth result in each first cycle; and
outputting the last delayed result in each second cycle.

186. The method for fast decoding of claim 185, wherein
at least one said multiplying comprises
an operation selected from a group consisting of:
shifting and bit remapping.
- 5
187. The method for fast decoding of claim 185, wherein
said inverse non-stationary cell filtering further comprises:
transforming the third result by first function N_1 to produce a seventh result;
transforming the fourth result by first function N_2 to produce an eight result;
10 adding the seventh result with the input sample to produce a fifth result; and
adding the eight result with last delayed result to produce a sixth result.
188. The method for fast decoding of claim 187, wherein
at least one said transforming comprises
15 an operation selected from a group consisting of:
shifting and bit remapping.
189. The method for fast decoding of claim 181, wherein
said inverse non-stationary cell filtering further comprises:
20 delaying an input sample for w samples to produce a delayed result in each first cycle;
second multiplying the delayed result with a second filter coefficient to produce a second
result;
first adding the second result with the input sample to produce a fourth result;
delaying the fourth result for w samples to produce the delayed result in each second cycle;
25 first multiplying the input sample with a first filter coefficient to produce a first result;
second adding the first result with the delayed result to produce a third result;
outputting the third result in each first cycle; and
outputting the delayed result in each second cycle.
- 30 190. The method for fast decoding of claim 189, wherein
at least one of said first multiplying and said second multiplying comprises

an operation selected from a group consisting of:
shifting and bit remapping.

5 191. The method for fast decoding of claim 190, wherein said operation comprises
shifting right for one bit position.

192. The method for fast decoding of claim 190, wherein said operation comprises
shifting right for two bit positions.

10 193. The method for fast decoding of claim 192, wherein:
said first adding comprises subtracting the second result from the input sample to produce
a fourth result; and
said second adding comprises subtracting the first result from the delayed result to produce
a third result.

15 194. The method for fast decoding of claim 181, wherein
said inverse non-stationary cell filtering further comprises:
delaying the input sample for w samples to produce a first delayed result in each first cycle;
delaying the first delayed result for w samples to produce a second delayed result;
20 delaying the second delayed result for w samples to produce a third delayed result;
first multiplying the input sample with a first filter coefficient to produce a first result;
second multiplying the first delayed result with a second filter coefficient to produce a
second result;
third multiplying the second delayed result with a third filter coefficient to produce a third
25 result;
fourth multiplying the third delayed result with a fourth filter coefficient to produce a
fourth result;
second adding the fourth result with the second result to produce a sixth result;
third adding the first result with the third result to produce a fifth result;
30 first adding the sixth result with the input sample to produce an eight result;
delaying the eight result for w samples to produce the first delayed result in each second
cycle;

fourth adding the fifth result with the third delayed result to produce a seventh result;
outputting the seventh result in each first cycle; and
outputting the third delayed result in each second cycle.

- 5 **195.** The method for fast decoding of claim 194, wherein
 at least one of said first multiplying, said second multiplying, said third multiplying and
said fourth multiplying **comprises**
 an operation selected from a group **consisting of**:
 shifting and bit remapping.
- 10 **196.** The method for fast decoding of claim 195, wherein said operation **comprises**
 shifting right for four bit positions.
- 197.** The method for fast decoding of claim 196, wherein:
15 said second adding **comprises** subtracting the fourth result from the second result to
produce a sixth result; and
 said third adding **comprises** subtracting the first result from the third result to produce a
fifth result.
- 20 **198.** The method for fast decoding of claim 168, wherein
 said estimating the probabilities of symbols within the specified contexts **comprises**
 updating adaptive histograms.
- 199.** The method for fast decoding of claim 198, wherein
25 said updating adaptive histograms **comprises**
 low-pass filtering probabilities selected from a group **consisting of**:
 probabilities of occurrences of a current symbol x ; and
 cumulative probabilities of occurrences of all symbols preceding said current symbol x .
- 30 **200.** The method for fast decoding of claim 199, wherein
 said updating adaptive histograms **further comprises**
 adapting a dominant pole during said low-pass filtering.

201. The method for fast decoding of claim 200, wherein
said adapting a dominant pole comprises
halving value of the dominant pole in each adaptation cycle.
- 5
202. The method for fast decoding of claim 168, wherein
said entropy decoding is range decoding, comprising
first multiplying a prescaled range r with a number $Q(x)$ selected from a group consisting
of:
10 a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.
203. The method for fast decoding of claim 202, wherein said first multiplying comprises:
15 simplified multiplying a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a
first result; and
shifting left the first result for l bit positions.
204. The method for fast decoding of claim 202, wherein said first multiplying comprises
20 shifting left said number $Q(x)$ for l bit positions.
205. The method for fast decoding of claim 202, wherein said first multiplying comprises:
zeroing a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shifting left said number $Q(x)$ for one bit position to produce a first result, when said small
25 number V is any odd number higher or equal 3;
adding said number $Q(x)$ with the first result to produce a second result; and
shifting left the second result for l bit positions.
206. The method for fast decoding of claim 202, wherein said first multiplying comprises:
30 zeroing a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;

shifting left said number $Q(x)$ for one bit position to produce a first result, when said small number V is equal 3;

shifting left said number $Q(x)$ for two bit positions to produce a first result, when said small number V is any odd number higher or equal 5;

5 adding said number $Q(x)$ with the first result to produce a second result; and
shifting left the second result for l bit positions.

207. The method for fast decoding of claim 168, wherein

said entropy decoding is range decoding, comprising

10 first dividing a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled range $r = \lfloor R/Total \rfloor$.

208. The method for fast decoding of claim 207, wherein said first dividing comprises

shifting right said range R for $w_3 = \log_2(Total)$ bit positions.

15

209. The method for fast decoding of claim 168, wherein

said entropy decoding is range decoding, comprising

second dividing a bottom range limit B with a prescaled range r , to produce a range correction $t = \lfloor B/r \rfloor$.

20

210. The method for fast decoding of claim 209, wherein said second dividing comprises:

simplified dividing said bottom range limit B with a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a first result; and

shifting right the first result for l bit positions.

25

211. The method for fast decoding of claim 209, wherein said second dividing comprises:

multiplying said bottom range limit B with a first predefined number, dependent on a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a first result; and

shifting right the first result for a sum of l and a second predefined number of bit positions,

30 dependent on said small number V .

212. The method for fast decoding of claim 168, wherein
said estimating the probabilities of symbols within the specified contexts **comprises**
reconstructing transformation coefficient C using a magnitude-set index MS , a sign S and a
residual R .

5

213. The method for fast decoding of claim 212, wherein
said entropy decoding **comprises**
decoding said residual R using inverse variable length coding (INVVLC).

10

214. The method for fast decoding of claim 212, wherein
said estimating the probabilities of symbols within the specified contexts **further**
comprises
context modeling of a transformation coefficient to be decoded, using already decoded
transformation coefficients.

15

215. The method for fast decoding of claim 214, wherein
said already decoded transformation coefficients are located north-east, north, north-west
and west from the transformation coefficient to be decoded.

20

216. The method for fast decoding of claim 214, wherein
said estimating the probabilities of symbols within the specified contexts **further**
comprises
determining a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said
already decoded transformation coefficients.

25

217. The method for fast decoding of claim 216, wherein
said estimating the probabilities of symbols within the specified contexts **further**
comprises

limiting a maximum mean value \overline{MS} by a constant ML to produce a magnitude context
30 MC .

218. The method for fast decoding of claim 217, wherein

said entropy decoding **comprises**

range decoding a magnitude-set index MS as a current symbol x , using an adaptive magnitude histogram $h[MC]$.

5

219. The fast decoding of claim 218, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

updating of an adaptive magnitude histogram $h[MC]$ using decoded said magnitude-set
10 index MS as a current symbol x .

220. The method for fast decoding of claim 214, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

15 determining a ternary context TC as the ternary code of sign values S_i of said already decoded transformation coefficients.

221. The method for fast decoding of claim 220, wherein

said estimating the probabilities of symbols within the specified contexts **further**
20 **comprises**

translating said ternary context TC into a sign context SC using CTX table.

222. The method for fast decoding of claim 221, wherein

said entropy decoding **comprises**

25 range decoding a sign S as a current symbol x , using an adaptive sign histogram $g[SC]$.

223. The method for fast decoding of claim 222, wherein

said estimating the probabilities of symbols within the specified contexts **further**
comprises

30 updating of an adaptive sign histogram $g[SC]$ using decoded said sign S as a current symbol x .

224. The method for fast decoding of claim 222, wherein

said estimating the probabilities of symbols within the specified contexts **further comprises**

5 inverting less probable decoded said sign *S* using NEG table.

225. An article of manufacture for fast encoding of input data into output compressed data **comprising** a storage medium with a machine readable code **causing the machine to:**

direct subband transform the input data to produce transformation coefficients;

10 estimate the probabilities of symbols within the specified contexts using the transformation coefficients to produce the probabilities of symbols within the specified contexts;

entropy encode transformation coefficients using the probabilities of symbols within the specified contexts to produce encoded data; and

substantially synchronize the encoded data to produce the output compressed data,

15 **whereby** machine performs lossless compression.

226. The article of manufacture for fast encoding of claim 225, **further causing the machine to** quantize the transformation coefficients to produce quantized transformation coefficients, **wherein:**

20 said step that estimates the probabilities of symbols within the specified contexts is performed using the quantized transformation coefficients to produce the probabilities of symbols within the specified contexts; and

said entropy encode quantized transformation coefficients is performed using the probabilities of symbols within the specified contexts to produce encoded data,

25 **whereby** machine performs lossy compression.

227. The article of manufacture for fast encoding of claim 225, **further causing the machine to** substantially synchronize the encoded data to produce synchronized compressed data, **wherein**

30 buffering the synchronized compressed data is performed to produce the output compressed data.

228. The article of manufacture for fast encoding of claim 225, **further causing the machine to**
color space convert original input data to produce input data.

229. The article of manufacture for fast encoding of claim 225, **wherein**

5 said step that direct subband transforms **causes the machine to:**

(a) direct subband transform the input data to produce transformation coefficients;

(b) direct subband transform selected transformation coefficients to produce transformed
transformation coefficients; and

(c) repeat step (b) finite number of times.

10

230. The article of manufacture for fast encoding of claim 229, **wherein** said selected
transformation coefficients are low-pass transformed for one-dimensional input data.

231. The article of manufacture for fast encoding of claim 229, **wherein** said selected
15 transformation coefficients are low-pass transformed both horizontally and vertically for two-
dimensional input data.

232. The article of manufacture for fast encoding of claim 225, **wherein**

20 said step that direct subband transforms **causes the machine to:**

direct filter horizontally; and

direct filter vertically.

233. The article of manufacture for fast encoding of claim 232, **wherein** said step that direct
filters horizontally is different from said step that direct filters vertically.

25

234. The article of manufacture for fast encoding of claim 232, **wherein**

at least one of said step that direct filters horizontally and said step that direct filters
vertically **causes the machine to**

direct non-stationary filter.

30

235. The article of manufacture for fast encoding of claim 225, wherein
said step that direct subband transforms **causes the machine to**
direct filter.
- 5 236. The article of manufacture for fast encoding of claim 235, wherein
said step that direct filters **causes the machine to**
direct non-stationary filter.
- 10 237. The article of manufacture for fast encoding of claim 236, wherein
said step that direct non-stationary filters **causes the machine to**
perform a plurality of successive direct non-stationary cell filterings.
- 15 238. The article of manufacture for fast encoding of claim 237, wherein
said step that direct non-stationary cell filters **causes the machine to:**
filter using first direct transfer function in the first cycle; and
filter using second direct transfer function in the second cycle.
- 20 239. The article of manufacture for fast encoding of claim 238, wherein
said first cycle is active during horizontal filtering of each second pixel; and
said second cycle is active during horizontal filtering of other pixels.
- 25 240. The article of manufacture for fast encoding of claim 238, wherein
said first cycle is active during vertical filtering of each second line; and
said second cycle is active during vertical filtering of other lines.
- 30 241. The article of manufacture for fast encoding of claim 238, wherein
said step that direct non-stationary filters **further causes the machine to:**
multiply a result of a plurality of steps that successive direct non-stationary cell filter with a
first gain number to produce a first result;
multiply a result of a plurality of steps that successive direct non-stationary cell filter with a
second gain number to produce a second result;

select the first result in each first cycle to produce an output sample; and
select the second result in each second cycle to produce the output sample.

242. The article of manufacture for fast encoding of claim 238, wherein

5 said step that direct non-stationary cell filters **further causes the machine to:**

 delay an input sample for w samples to produce a plurality of even and odd indexed
delayed results in each first cycle;

 multiply each even indexed delayed result with appropriate first filter coefficient selected
from a group of first filter coefficients to produce first results;

10 add together all first results to produce a third result;

 add the third result with the input sample to produce a fifth result;

 delay the fifth result for w samples to produce a plurality of even and odd indexed delayed
results in each second cycle;

15 multiply the input sample and each odd indexed delayed result with appropriate second
filter coefficient selected from a group of second filter coefficients to produce second results;

 add together all second results to produce a fourth result;

 add the fourth result with last delayed result to produce a sixth result;

 output the sixth result in each first cycle; and

 output the last delayed result in each second cycle.

20

243. The article of manufacture for fast encoding of claim 242, wherein

 at least one said step that multiplies **causes the machine to**

 perform an operation selected from a group **consisting of:**

 shift and bit remap.

25

244. The article of manufacture for fast encoding of claim 242, wherein

 said step that direct non-stationary filters **further causes the machine to:**

 transform the third result by first function N_1 to produce a seventh result;

 transform the fourth result by first function N_2 to produce an eight result;

30 add the seventh result with the input sample to produce a fifth result; and

 add the eight result with last delayed result to produce a sixth result.

245. The article of manufacture for fast encoding of claim 244, wherein
at least one said step that transforms **causes the machine to**
perform an operation selected from a group consisting of:
shift and bit remap.

5

246. The article of manufacture for fast encoding of claim 238, wherein
said step that direct non-stationary cell filters **further causes the machine to:**
delay an input sample for w samples to produce a delayed result in each first cycle;
multiply the delayed result with a second filter coefficient to produce a second result;
10 add the second result with the input sample to produce a fourth result;
delay the fourth result for w samples to produce the delayed result in each second cycle;
multiply the input sample with a first filter coefficient to produce a first result;
add the first result with the delayed result to produce a third result;
output the third result in each first cycle; and
15 output the delayed result in each second cycle.

15

247. The article of manufacture for fast encoding of claim 246, wherein
at least one said step that multiplies **causes the machine to**
perform an operation selected from a group consisting of:
20 shift and bit remap.

20

248. The article of manufacture for fast encoding of claim 247, wherein
said operation **causes the machine to**
shift right data for two bit positions.

25

249. The article of manufacture for fast encoding of claim 247, wherein
said operation **causes the machine to**
shift right data for one bit position.

30

250. The article of manufacture for fast encoding of claim 249, wherein
said step that adds the second result **causes the machine to** subtract the second result from
the input sample to produce a fourth result; and

said step that adds the first result **causes the machine to** subtract the first result from the delayed result to produce a third result.

251. The article of manufacture for fast encoding of claim 238, wherein

5 said step that direct non-stationary cell filters **further causes the machine to:**
 delay the input sample for w samples to produce a first delayed result in each first cycle;
 delay the first delayed result for w samples to produce a second delayed result;
 delay the second delayed result for w samples to produce a third delayed result;
 multiply the input sample with a first filter coefficient to produce a first result;
10 multiply the first delayed result with a second filter coefficient to produce a second result;
 multiply the second delayed result with a third filter coefficient to produce a third result;
 multiply the third delayed result with a fourth filter coefficient to produce a fourth result;
 add the second result with the fourth result to produce a sixth result;
 add the third result with the first result to produce a fifth result;
15 add the sixth result with the input sample to produce an eight result;
 delay the eight result for w samples to produce the first delayed result in each second cycle;
 add the fifth result with the third delayed result to produce a seventh result;
 output the seventh result in each first cycle; and
 output the third delayed result in each second cycle.

20

252. The article of manufacture for fast encoding of claim 251, wherein

 at least one said step that multiplies **causes the machine to**
 perform an operation selected from a group **consisting of:**
 shift and bit remap.

25

253. The article of manufacture for fast encoding of claim 252, wherein

 said operation **causes the machine to**
 shift right data for four bit positions.

30 254. The article of manufacture for fast encoding of claim 253, wherein:

 said step that adds the second result **causes the machine to** subtract the second result from the fourth result to produce a sixth result; and

said step that adds the third result **causes the machine** to subtract the third result from the first result to produce a fifth result.

255. The article of manufacture for fast encoding of claim 225, wherein

- 5 said step that estimates the probabilities of symbols within the specified contexts **causes the machine to**
update adaptive histograms.

256. The article of manufacture for fast encoding of claim 255, wherein

- 10 said step that updates adaptive histograms **causes the machine to**
low-pass filter probabilities selected from a group **consisting of**:
probabilities of occurrences of a current symbol x ; and
cumulative probabilities of occurrences of all symbols preceding the current symbol x .

257. The article of manufacture for fast encoding of claim 256, wherein

- 15 said step that updates adaptive histograms **further causes the machine to**
adapt a dominant pole during said low-pass filtering.

258. The article of manufacture for fast encoding of claim 257, wherein

- 20 said step that adapts a dominant pole **causes the machine to**
halve value of the dominant pole in each adaptation cycle.

259. The article of manufacture for fast encoding of claim 225, wherein

- 25 said step that entropy encodes is a step that range encode, **causing the machine to**
multiply a prescaled range r with a number $Q(x)$ selected from a group **consisting of**:
a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.

260. The article of manufacture for fast encoding of claim 259, wherein

- 30 said step that multiplies **causes the machine to**:

simplified multiply a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a first result; and

shift left the first result for l bit positions.

5 261. The article of manufacture for fast encoding of claim 259, wherein
said step that multiplies **causes the machine to**
shift left said number $Q(x)$ for l bit positions.

10 262. The article of manufacture for fast encoding of claim 259, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is any odd number higher or equal 3;
add said number $Q(x)$ with the first result to produce a second result; and
15 shift left the second result for l bit positions.

20 263. The article of manufacture for fast encoding of claim 259, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is equal 3;
shift left said number $Q(x)$ for two bit positions to produce a first result, when said small
number V is any odd number higher or equal 5;
add said number $Q(x)$ with the first result to produce a second result; and
25 shift left the second result for l bit positions.

30 264. The article of manufacture for fast encoding of claim 225, wherein
said step that entropy encodes is a step that range encode, **causing the machine to:**
divide a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled
range $r = \lfloor R/Total \rfloor$.

265. The article of manufacture for fast encoding of claim 264, wherein

said step that divides **causes the machine to**
shift right range R for $w_3 = \log_2(Total)$ bit positions.

5

266. The article of manufacture for fast encoding of claim 225, wherein

said step that estimates the probabilities of symbols within the specified contexts **causes**
the machine to:

split a transformation coefficient C into a sign S and a magnitude M ;
10 determine a magnitude-set index MS using said magnitude M and a magnitude-set table;
and
determine a residual R using said magnitude M and said magnitude-set table.

267. The article of manufacture for fast encoding of claim 266, wherein

15 said magnitude-set index MS is determined to be equal to a sum of a doubled position of
the first nonzero bit of the highest significance and the value of the first next bit of the lower
significance in a binary representation of said magnitude M ; and

said residual R is determined as the difference between said magnitude M and the lower
coefficient limit, equal to a value of said magnitude M with all bits zeroed except the first
20 nonzero bit of the highest significance and the first next bit of the lower significance in a binary
representation of said magnitude M .

268. The article of manufacture for fast encoding of claim 266, wherein

said step that entropy encodes **causes the machine to**
25 encode a residual R using variable length coding (VLC).

269. The article of manufacture for fast encoding of claim 266, wherein

said estimates the probabilities of symbols within the specified contexts **further causes the**
machine to
30 context model a transformation coefficient to be encoded, using already encoded
transformation coefficients.

270. The article of manufacture for fast encoding of claim 269, wherein

said already encoded transformation coefficients are located north-east, north, north-west and west from said transformation coefficient to be encoded.

5 271. The article of manufacture for fast encoding of claim 269, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

determine a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said already encoded transformation coefficients.

10

272. The article of manufacture for fast encoding of claim 271, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

limit a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .

15

273. The article of manufacture for fast encoding of claim 272, wherein

said step that entropy encodes **causes the machine to**

range encode said magnitude-set index MS as a current symbol x , using said adaptive magnitude histogram $h[MC]$.

20

274. The article of manufacture for fast encoding of claim 272, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

update an adaptive magnitude histogram $h[MC]$ using said magnitude-set index MS as a

25

current symbol x .

275. The article of manufacture for fast encoding of claim 269, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

30

determine a ternary context TC as the ternary code of sign values S_i of said already encoded transformation coefficients.

276. The article of manufacture for fast encoding of claim 275, **wherein**
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

5 invert less probable said sign S using NEG table.

277. The article of manufacture for fast encoding of claim 275, **wherein**
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

10 translate said ternary context TC into a sign context SC using CTX table.

278. The article of manufacture for fast encoding of claim 277, **wherein**
said step that entropy encodes **causes the machine to**
range encode said sign S as a current symbol x , using said adaptive sign histogram $g[SC]$.

15

279. The article of manufacture for fast encoding of claim 277, **wherein**
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

update of an adaptive sign histogram $g[SC]$ using said sign S as a current symbol x .

20

280. An article of manufacture for fast decoding of input compressed data into output data
comprising a storage medium with a machine readable code **causing the machine to:**
substantially synchronize the input compressed data to produce synchronized compressed
data;

25 entropy decode the synchronized compressed data using the probabilities of symbols within
the specified contexts to produce transformation coefficients;

estimate the probabilities of symbols within the specified contexts using the transformation
coefficients to produce the probabilities of symbols within the specified contexts; and

inverse subband transform the transformation coefficients to produce the output data,

30 **whereby** machine performs lossless decompression.

281. The article of manufacture for fast decoding of claim 280, further causing the machine to dequantize transformation coefficients to produce dequantized transformation coefficients, wherein

inverse subband transforming the dequantized transformation coefficients is performed to produce the output data,
5 wherein machine performs lossy decompression.

282. The article of manufacture for fast decoding of claim 280, further causing the machine to buffer the input compressed data to produce buffered compressed data,

10 wherein substantially synchronizing the buffered compressed data is performed to produce the synchronized compressed data.

283. The article of manufacture for fast decoding of claim 280, further causing the machine to color space convert the output data to produce converted output data.
15

284. The article of manufacture for fast decoding of claim 280, wherein said step that inverse subband transforms causes the machine to:

- (a) inverse subband transform transformation coefficients to produce selected transformation coefficients;
20 (b) repeat step (a) finite number of times; and
(c) inverse subband transform transformation coefficients to produce the output data.

285. The article of manufacture for fast decoding of claim 284, wherein said selected transformation coefficients are low-pass transformed for one-dimensional input data.
25

286. The article of manufacture for fast decoding of claim 284, wherein said selected transformation coefficients are low-pass transformed both horizontally and vertically for two-dimensional input data.

30

287. The article of manufacture for fast decoding of claim 280, wherein said step that inverse subband transforms causes the machine to:

inverse filter horizontally; and
inverse filter vertically.

288. The article of manufacture for fast decoding of claim 287, wherein said step that inverse
5 filters horizontally is different from said step that inverse filters vertically.

289. The article of manufacture for fast decoding of claim 287, wherein
at least one of said step that inverse filters horizontally and said step that inverse filters
vertically **causes the machine to**
10 inverse non-stationary filter.

290. The article of manufacture for fast decoding of claim 280, wherein
said step that inverse subband transforms **causes the machine to**
inverse filter.
15

291. The article of manufacture for fast decoding of claim 290, wherein
said step that inverse filters **causes the machine to**
inverse non-stationary filter.

20 292. The article of manufacture for fast decoding of claim 291, wherein
said step that inverse non-stationary filters **causes the machine to**
perform a plurality of successive inverse non-stationary cell filterings.

293. The article of manufacture for fast decoding of claim 292, wherein
25 said step that inverse non-stationary cell filters **causes the machine to:**
filter using first direct transfer function in the first cycle; and
filter using second direct transfer function in the second cycle.

294. The article of manufacture for fast decoding of claim 293, wherein
30 said second cycle is active during horizontal filtering of each second pixel; and
said first cycle is active during horizontal filtering of other pixels.

295. The article of manufacture for fast decoding of claim 293, wherein
said second cycle is active during vertical filtering of each second line; and
said first cycle is active during vertical filtering of other lines.

5 296. The article of manufacture for fast decoding of claim 293, wherein
said step that inverse non-stationary filters further causes the machine to:
multiply an input with a reciprocal value of a first gain number to produce a first result;
multiply an input with a reciprocal value of a second gain number to produce a second
result;
10 select the first result in each second cycle to produce an input sample for a plurality of steps
that successive inverse non-stationary cell filter; and
select the second result in each first cycle to produce an input sample for a plurality of steps
that successive inverse non-stationary cell filter.

15 297. The article of manufacture for fast decoding of claim 293, wherein
said step that inverse non-stationary cell filters further causes the machine to:
delay an input sample for w samples to produce a plurality of even and odd indexed
delayed results in each first cycle;
multiply each even indexed delayed result with appropriate first filter coefficient selected
20 from a group of first filter coefficients to produce first results;
add together all first results to produce a third result;
add the third result with the input sample to produce a fifth result;
delay the fifth result for w samples to produce a plurality of even and odd indexed delayed
results in each second cycle;
25 multiply the input sample and each odd indexed delayed result with appropriate second
filter coefficient selected from a group of second filter coefficients to produce second results;
add together all second results to produce a fourth result;
add the fourth result with last delayed result to produce a sixth result;
output the sixth result in each first cycle; and
30 output the last delayed result in each second cycle.

298. The article of manufacture for fast decoding of claim 297, wherein
at least one said step that multiplies causes the machine to
perform an operation selected from a group consisting of:
shift and bit remap.

5

299. The article of manufacture for fast decoding of claim 297, wherein
said step that inverse non-stationary cell filters further causes the machine to
transform the third result by a first function N_1 to produce a seventh result;
transform the fourth result by a first function N_2 to produce an eight result;
add the seventh result with the input sample to produce a fifth result; and
add the eight result with last delayed result to produce a sixth result.

10

300. The article of manufacture for fast decoding of claim 299, wherein
at least one said step that transforms causes the machine to
perform an operation selected from a group consisting of:
shift and bit remap.

15

301. The article of manufacture for fast decoding of claim 293, wherein
said step that inverse non-stationary cell filters further causes the machine to:
delay an input sample for w samples to produce a delayed result in each first cycle;
multiply the delayed result with a second filter coefficient to produce a second result;
add the second result with the input sample to produce a fourth result;
delay the fourth result for w samples to produce the delayed result in each second cycle;
multiply the input sample with a first filter coefficient to produce a first result;
add the first result with the delayed result to produce a third result;
output the third result in each first cycle; and
output the delayed result in each second cycle.

20

25

302. The article of manufacture for fast decoding of claim 301, wherein
at least one said step that multiplies causes the machine to
perform an operation selected from a group consisting of:
shift and bit remap.

30

303. The article of manufacture for fast decoding of claim 302, wherein
said operation **causes the machine to**
shift right for one bit position.

5

304. The article of manufacture for fast decoding of claim 302, wherein
said operation **causes the machine to**
shift right data for two bit positions.

10 305. The article of manufacture for fast decoding of claim 304, wherein
said step that adds the second result **causes the machine to** subtract the second result from
the input sample to produce a fourth result; and
said step that adds the first result **causes the machine to** subtract the first result from the
delayed result to produce a third result.

15

306. The article of manufacture for fast decoding of claim 293, wherein
said step that inverse non-stationary cell filters **further causes the machine to:**
delay an input sample for w samples to produce a first delayed result in each first cycle;
delay the first delayed result for w samples to produce a second delayed result;
20 delay the second delayed result for w samples to produce a third delayed result;
multiply the input sample with a first filter coefficient to produce a first result;
multiply the first delayed result with a second filter coefficient to produce a second result;
multiply the second delayed result with a third filter coefficient to produce a third result;
multiply the third delayed result with a fourth filter coefficient to produce a fourth result;
25 add the fourth result with the second result to produce a sixth result;
add the first result with the third result to produce a fifth result;
add the sixth result with the input sample to produce an eight result;
delay the eight result for w samples to produce the first delayed result in each second cycle;
add the fifth result with the third delayed result to produce a seventh result;
30 output the seventh result in each first cycle; and
output the third delayed result in each second cycle.

307. The article of manufacture for fast decoding of claim 306, wherein
at least one said step that multiplies **causes the machine to**
perform an operation selected from a group consisting of:
shift and bit remap.

5

308. The article of manufacture for fast decoding of claim 307, wherein
said operation **causes the machine to**
shift right data for four bit positions.

10 309. The article of manufacture for fast decoding of claim 308, wherein
said step that adds the fourth result **causes the machine to** subtract the fourth result from
the second result to produce a sixth result; and
said step that adds the first result **causes the machine to** subtract the first result from the
third result to produce a fifth result.

15

310. The article of manufacture for fast decoding of claim 280, wherein
said step that estimates the probabilities of symbols within the specified contexts **causes**
the machine to
update adaptive histograms.

20

311. The article of manufacture for fast decoding of claim 310, wherein
said step that updates adaptive histograms **causes the machine to**
low-pass filter probabilities selected from a group consisting of:
probabilities of occurrences of a current symbol x ; and
25 cumulative probabilities of occurrences of all symbols preceding said current symbol x .

312. The article of manufacture for fast decoding of claim 311, wherein
said step that updates adaptive histograms **further causes the machine to**
adapt a dominant pole during said low-pass filtering.

30

313. The article of manufacture for fast decoding of claim 312, wherein
said step that adapts a dominant pole **causes the machine to**
halve value of the dominant pole in each adaptation cycle.

5 314. The article of manufacture for fast decoding of claim 280, wherein
said step that entropy decodes is a step that range decode, **causing the machine to**
multiply a prescaled range r with a number $Q(x)$ selected from a group consisting of:
a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
10 a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.

315. The article of manufacture for fast decoding of claim 314, wherein
said step that multiplies **causes the machine to:**
simplified multiply a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a first
15 result; and
shift left the first result for l bit positions.

316. The article of manufacture for fast decoding of claim 314, wherein
said step that multiplies **causes the machine to**
20 shift left said number $Q(x)$ for l bit positions.

317. The article of manufacture for fast decoding of claim 314, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
25 shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is any odd number higher or equal 3;
add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.

30 318. The article of manufacture for fast decoding of claim 314, wherein
said step that multiplies **causes the machine to:**

zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;

shift left said number $Q(x)$ for one bit position to produce a first result, when said small number V is equal 3;

shift left said number $Q(x)$ for two bit positions to produce a first result, when said small number V is any odd number higher or equal 5;
add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.

319. The article of manufacture for fast decoding of claim 280, wherein
said step that entropy decodes is a step that range decode, causing the machine to
divide a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled
range $r = \lfloor R/Total \rfloor$.

320. The article of manufacture for fast decoding of claim 319, wherein
said step that divides causes the machine to
shift right said range R for $w_3 = \log_2(Total)$ bit positions.

321. The article of manufacture for fast decoding of claim 280, wherein
said step that entropy decodes is a step that range decode, causing the machine to
divide a bottom range limit B with a prescaled range r , to produce a range correction
 $t = \lfloor B/r \rfloor$.

322. The article of manufacture for fast decoding of claim 321, wherein
said step that divides causes the machine to:
simplified divide said bottom range limit B with a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a
first result; and
shift right the first result for l bit positions.

323. The article of manufacture for fast decoding of claim 321, wherein
said step that divides causes the machine to:

multiply said bottom range limit B with a first predefined number, dependent on a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a first result; and

shift right the first result for a sum of l and a second predefined number of bit positions, dependent on said small number V .

5

324. The article of manufacture for fast decoding of claim 280, wherein

said step that estimates the probabilities of symbols within the specified contexts **causes the machine to**

reconstruct transformation coefficient C , using a magnitude-set index MS , a sign S and a residual R .

10

325. The article of manufacture for fast decoding of claim 324, wherein

said step that entropy decodes **causes the machine to**

decode said residual R using inverse variable length coding (INVVLC).

15

326. The article of manufacture for fast decoding of claim 324, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

context model a transformation coefficient to be decoded, using already decoded transformation coefficients.

20

327. The article of manufacture for fast decoding of claim 326, wherein

said already decoded transformation coefficients are located north-east, north, north-west and west from the transformation coefficient to be decoded.

25

328. The article of manufacture for fast decoding of claim 326, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

determine a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said already decoded transformation coefficients.

30

329. The article of manufacture for fast decoding of claim 328, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

limit a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .

5

330. The article of manufacture for fast decoding of claim 329, wherein
said step that entropy decodes **causes the machine to**
range decode said magnitude-set index MS as a current symbol x , using said adaptive
magnitude histogram $h[MC]$.

10

331. The article of manufacture for fast decoding of claim 330, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

update an adaptive magnitude histogram $h[MC]$ using said magnitude-set index MS as a

15

current symbol x .

332. The article of manufacture for fast decoding of claim 326, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

20

determine a ternary context TC as the ternary code of sign values S_i of said already
decoded transformation coefficients.

333. The article of manufacture for fast decoding of claim 332, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to

25

translate said ternary context TC into a sign context SC using CTX table.

334. The article of manufacture for fast decoding of claim 333, wherein
said step that entropy decodes **causes the machine to**

30

range decode sign S as a current symbol x , using an adaptive sign histogram $g[SC]$.

335. The article of manufacture for fast decoding of claim 334, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to
update an adaptive sign histogram $g[SC]$ using decoded sign S as a current symbol x .

5

336. The article of manufacture for fast decoding of claim 334, wherein
said step that estimates the probabilities of symbols within the specified contexts **further**
causes the machine to
invert less probable decoded sign S using NEG table.

10

337. A data signal for fast encoding of input data into output compressed data embodied in a
carrier wave **comprising** a machine readable code **causing the machine to:**
direct subband transform the input data to produce transformation coefficients;
estimate the probabilities of symbols within the specified contexts using the transformation
coefficients to produce the probabilities of symbols within the specified contexts;
entropy encode transformation coefficients using the probabilities of symbols within the
specified contexts to produce encoded data; and
substantially synchronize the encoded data to produce the output compressed data,
whereby machine performs lossless compression.

20

338. The data signal for fast encoding of claim 337, **further causing the machine to**
quantize the transformation coefficients to produce quantized transformation coefficients,
wherein:
said step that estimates the probabilities of symbols within the specified contexts is
performed using the quantized transformation coefficients to produce the probabilities of
symbols within the specified contexts; and
said entropy encode quantized transformation coefficients is performed using the
probabilities of symbols within the specified contexts to produce encoded data,
whereby machine performs lossy compression.

30

339. The data signal for fast encoding of claim 337, **further causing the machine to**
substantially synchronize the encoded data to produce synchronized compressed data,
wherein
buffering the synchronized compressed data is performed to produce the output
5 compressed data.
340. The data signal for fast encoding of claim 337, **further causing the machine to**
color space convert original input data to produce input data.
- 10 341. The data signal for fast encoding of claim 337, **wherein**
said step that direct subband transforms **causes the machine to:**
(a) direct subband transform the input data to produce transformation coefficients;
(b) direct subband transform selected transformation coefficients to produce transformed
transformation coefficients; and
15 (c) repeat step (b) finite number of times.
342. The data signal for fast encoding of claim 341, **wherein** said selected transformation
coefficients are low-pass transformed for one-dimensional input data.
- 20 343. The data signal for fast encoding of claim 341, **wherein** said selected transformation
coefficients are low-pass transformed both horizontally and vertically for two-dimensional input
data.
344. The data signal for fast encoding of claim 337, **wherein**
25 said step that direct subband transforms **causes the machine to:**
direct filter horizontally; and
direct filter vertically.
345. The data signal for fast encoding of claim 344, **wherein** said step that direct filters
30 horizontally is different from said step that direct filters vertically.

346. The data signal for fast encoding of claim 344, wherein
at least one of said step that direct filters horizontally and said step that direct filters
vertically **causes the machine to**
direct non-stationary filter.

5

347. The data signal for fast encoding of claim 337, wherein
said step that direct subband transforms **causes the machine to**
direct filter.

10 348. The data signal for fast encoding of claim 347, wherein
said step that direct filters **causes the machine to**
direct non-stationary filter.

15 349. The data signal for fast encoding of claim 348, wherein
said step that direct non-stationary filters **causes the machine to**
perform a plurality of successive direct non-stationary cell filterings.

20 350. The data signal for fast encoding of claim 349, wherein
said step that direct non-stationary cell filters **causes the machine to:**
filter using first direct transfer function in the first cycle; and
filter using second direct transfer function in the second cycle.

25 351. The data signal for fast encoding of claim 350, wherein
said first cycle is active during horizontal filtering of each second pixel; and
said second cycle is active during horizontal filtering of other pixels.

30 352. The data signal for fast encoding of claim 350, wherein
said first cycle is active during vertical filtering of each second line; and
said second cycle is active during vertical filtering of other lines.

353. The data signal for fast encoding of claim 350, wherein
said step that direct non-stationary filters **further causes the machine to:**

multiply a result of a plurality of steps that successive direct non-stationary cell filter with a first gain number to produce a first result;

multiply a result of a plurality of steps that successive direct non-stationary cell filter with a second gain number to produce a second result;

- 5 select the first result in each first cycle to produce an output sample; and
 select the second result in each second cycle to produce the output sample.

354. The data signal for fast encoding of claim 350, wherein

 said step that direct non-stationary cell filters **further causes the machine to:**

- 10 delay an input sample for w samples to produce a plurality of even and odd indexed delayed results in each first cycle;

 multiply each even indexed delayed result with appropriate first filter coefficient selected from a group of first filter coefficients to produce first results;

 add together all first results to produce a third result;

- 15 add the third result with the input sample to produce a fifth result;

 delay the fifth result for w samples to produce a plurality of even and odd indexed delayed results in each second cycle;

 multiply the input sample and each odd indexed delayed result with appropriate second filter coefficient selected from a group of second filter coefficients to produce second results;

- 20 add together all second results to produce a fourth result;

 add the fourth result with last delayed result to produce a sixth result;

 output the sixth result in each first cycle; and

 output the last delayed result in each second cycle.

- 25 355. The data signal for fast encoding of claim 354, wherein

 at least one said step that multiplies **causes the machine to**
 perform an operation selected from a group consisting of:
 shift and bit remap.

- 30 356. The data signal for fast encoding of claim 354, wherein

 said step that direct non-stationary filters **further causes the machine to:**
 transform the third result by first function N_1 to produce a seventh result;

transform the fourth result by first function N_2 to produce an eight result;
add the seventh result with the input sample to produce a fifth result; and
add the eight result with last delayed result to produce a sixth result.

5 **357.** The data signal for fast encoding of claim 356, wherein

at least one said step that transforms **causes the machine to**
perform an operation selected from a group **consisting of:**
shift and bit remap.

10 **358.** The data signal for fast encoding of claim 350, wherein

said step that direct non-stationary cell filters **further causes the machine to:**
delay an input sample for w samples to produce a delayed result in each first cycle;
multiply the delayed result with a second filter coefficient to produce a second result;
add the second result with the input sample to produce a fourth result;
15 delay the fourth result for w samples to produce the delayed result in each second cycle;
multiply the input sample with a first filter coefficient to produce a first result;
add the first result with the delayed result to produce a third result;
output the third result in each first cycle; and
output the delayed result in each second cycle.

20

359. The data signal for fast encoding of claim 358, wherein

at least one said step that multiplies **causes the machine to**
perform an operation selected from a group **consisting of:**
shift and bit remap.

25

360. The data signal for fast encoding of claim 359, wherein

said operation **causes the machine to**
shift right data for two bit positions.

30 **361.** The data signal for fast encoding of claim 359, wherein

said operation **causes the machine to**
shift right data for one bit position.

362. The data signal for fast encoding of claim 361, wherein

said step that adds the second result **causes the machine** to subtract the second result from the input sample to produce a fourth result; and

5 said step that adds the first result **causes the machine** to subtract the first result from the delayed result to produce a third result.

363. The data signal for fast encoding of claim 350, wherein

said step that direct non-stationary cell filters **further causes the machine to:**

10 delay the input sample for w samples to produce a first delayed result in each first cycle;
 delay the first delayed result for w samples to produce a second delayed result;
 delay the second delayed result for w samples to produce a third delayed result;
 multiply the input sample with a first filter coefficient to produce a first result;
 multiply the first delayed result with a second filter coefficient to produce a second result;
15 multiply the second delayed result with a third filter coefficient to produce a third result;
 multiply the third delayed result with a fourth filter coefficient to produce a fourth result;
 add the second result with the fourth result to produce a sixth result;
 add the third result with the first result to produce a fifth result;
 add the sixth result with the input sample to produce an eight result;
20 delay the eight result for w samples to produce the first delayed result in each second cycle;
 add the fifth result with the third delayed result to produce a seventh result;
 output the seventh result in each first cycle; and
 output the third delayed result in each second cycle.

25 364. The data signal for fast encoding of claim 363, wherein

at least one said step that multiplies **causes the machine** to
perform an operation selected from a group **consisting of:**
shift and bit remap.

30 365. The data signal for fast encoding of claim 364, wherein

said operation **causes the machine** to
shift right data for four bit positions.

366. The data signal for fast encoding of claim 365, wherein:

said step that adds the second result **causes the machine to** subtract the second result from the fourth result to produce a sixth result; and

5 said step that adds the third result **causes the machine to** subtract the third result from the first result to produce a fifth result.

367. The data signal for fast encoding of claim 337, wherein

10 said step that estimates the probabilities of symbols within the specified contexts **causes the machine to**
update adaptive histograms.

368. The data signal for fast encoding of claim 367, wherein

15 said step that updates adaptive histograms **causes the machine to**
low-pass filter probabilities selected from a group **consisting of:**
probabilities of occurrences of a current symbol x ; and
cumulative probabilities of occurrences of all symbols preceding the current symbol x .

369. The data signal for fast encoding of claim 368, wherein

20 said step that updates adaptive histograms **further causes the machine to**
adapt a dominant pole during said low-pass filtering.

370. The data signal for fast encoding of claim 369, wherein

25 said step that adapts a dominant pole **causes the machine to**
halve value of the dominant pole in each adaptation cycle.

371. The data signal for fast encoding of claim 337, wherein

30 said step that entropy encodes is a step that range encode, **causing the machine to**
multiply a prescaled range r with a number $Q(x)$ selected from a group **consisting of:**
a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.

372. The data signal for fast encoding of claim 371, wherein
said step that multiplies **causes the machine to:**
simplified multiply a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a first
5 result; and
shift left the first result for l bit positions.
373. The data signal for fast encoding of claim 371, wherein
said step that multiplies **causes the machine to**
10 shift left said number $Q(x)$ for l bit positions.
374. The data signal for fast encoding of claim 371, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
15 shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is any odd number higher or equal 3;
add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.
- 20 375. The data signal for fast encoding of claim 371, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is equal 3;
25 shift left said number $Q(x)$ for two bit positions to produce a first result, when said small
number V is any odd number higher or equal 5;
add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.
- 30 376. The data signal for fast encoding of claim 337, wherein
said step that entropy encodes is a step that range encode, **causing the machine to:**

divide a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled range $r = \lfloor R/Total \rfloor$.

377. The data signal for fast encoding of claim 376, wherein
5 said step that divides **causes the machine to**
shift right range R for $w_3 = \log_2(Total)$ bit positions.

378. The data signal for fast encoding of claim 337, wherein
said step that estimates the probabilities of symbols within the specified contexts **causes**
10 **the machine to:**
split a transformation coefficient C into a sign S and a magnitude M ;
determine a magnitude-set index MS using said magnitude M and a magnitude-set table;
and
determine a residual R using said magnitude M and said magnitude-set table.

15
379. The data signal for fast encoding of claim 378, wherein
said magnitude-set index MS is determined to be equal to a sum of a doubled position of
the first nonzero bit of the highest significance and the value of the first next bit of the lower
significance in a binary representation of said magnitude M ; and
20 said residual R is determined as the difference between said magnitude M and the lower
coefficient limit, equal to a value of said magnitude M with all bits zeroed except the first
nonzero bit of the highest significance and the first next bit of the lower significance in a binary
representation of said magnitude M .

25 380. The data signal for fast encoding of claim 378, wherein
said step that entropy encodes **causes the machine to**
encode a residual R using variable length coding (VLC).

381. The data signal for fast encoding of claim 378, wherein
30 said estimates the probabilities of symbols within the specified contexts **further causes the**
machine to

context model a transformation coefficient to be encoded, using already encoded transformation coefficients.

382. The data signal for fast encoding of claim 381, wherein

5 said already encoded transformation coefficients are located north-east, north, north-west and west from said transformation coefficient to be encoded.

383. The data signal for fast encoding of claim 381, wherein

10 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

 determine a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said already encoded transformation coefficients.

384. The data signal for fast encoding of claim 383, wherein

15 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

 limit a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .

385. The data signal for fast encoding of claim 384, wherein

20 said step that entropy encodes **causes the machine to**

 range encode said magnitude-set index MS as a current symbol x , using said adaptive magnitude histogram $h[MC]$.

386. The data signal for fast encoding of claim 384, wherein

25 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

 update an adaptive magnitude histogram $h[MC]$ using said magnitude-set index MS as a current symbol x .

30 387. The data signal for fast encoding of claim 381, wherein

 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

determine a ternary context TC as the ternary code of sign values S_i of said already encoded transformation coefficients.

388. The data signal for fast encoding of claim 387, wherein

- 5 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**
 invert less probable said sign S using NEG table.

389. The data signal for fast encoding of claim 387, wherein

- 10 said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**
 translate said ternary context TC into a sign context SC using CTX table.

390. The data signal for fast encoding of claim 389, wherein

- 15 said step that entropy encodes **causes the machine to**
 range encode said sign S as a current symbol x , using said adaptive sign histogram $g[SC]$.

391. The data signal for fast encoding of claim 389, wherein

- said step that estimates the probabilities of symbols within the specified contexts **further**
20 **causes the machine to**
 update of an adaptive sign histogram $g[SC]$ using said sign S as a current symbol x .

392. A data signal for fast decoding of input compressed data into output data embodied in a carrier wave **comprising** a machine readable code **causing the machine to:**

- 25 substantially synchronize the input compressed data to produce synchronized compressed data;
 entropy decode the synchronized compressed data using the probabilities of symbols within the specified contexts to produce transformation coefficients;
 estimate the probabilities of symbols within the specified contexts using the transformation
30 coefficients to produce the probabilities of symbols within the specified contexts; and
 inverse subband transform the transformation coefficients to produce the output data,
whereby machine performs lossless decompression.

393. The data signal for fast decoding of claim 392, further causing the machine to
dequantize transformation coefficients to produce dequantized transformation coefficients,
wherein
5 inverse subband transforming the dequantized transformation coefficients is performed to
produce the output data,
whereby machine performs lossy decompression.
394. The data signal for fast decoding of claim 392, further causing the machine to
10 buffer the input compressed data to produce buffered compressed data,
wherein
substantially synchronizing the buffered compressed data is performed to produce the
synchronized compressed data.
- 15 395. The data signal for fast decoding of claim 392, further causing the machine to
color space convert the output data to produce converted output data.
396. The data signal for fast decoding of claim 392, wherein
said step that inverse subband transforms causes the machine to:
20 (a) inverse subband transform transformation coefficients to produce selected
transformation coefficients;
 (b) repeat step (a) finite number of times; and
 (c) inverse subband transform transformation coefficients to produce the output data.
- 25 397. The data signal for fast decoding of claim 396, wherein said selected transformation
coefficients are low-pass transformed for one-dimensional input data.
398. The data signal for fast decoding of claim 396, wherein said selected transformation
coefficients are low-pass transformed both horizontally and vertically for two-dimensional input
30 data.

399. The data signal for fast decoding of claim 392, wherein
said step that inverse subband transforms causes the machine to:
inverse filter horizontally; and
inverse filter vertically.

5

400. The data signal for fast decoding of claim 399, wherein said step that inverse filters
horizontally is different from said step that inverse filters vertically.

10

401. The data signal for fast decoding of claim 399, wherein
at least one of said step that inverse filters horizontally and said step that inverse filters
vertically causes the machine to
inverse non-stationary filter.

15

402. The data signal for fast decoding of claim 392, wherein
said step that inverse subband transforms causes the machine to
inverse filter.

20

403. The data signal for fast decoding of claim 402, wherein
said step that inverse filters causes the machine to
inverse non-stationary filter.

25

404. The data signal for fast decoding of claim 403, wherein
said step that inverse non-stationary filters causes the machine to
perform a plurality of successive inverse non-stationary cell filterings.

30

405. The data signal for fast decoding of claim 404, wherein
said step that inverse non-stationary cell filters causes the machine to:
filter using first direct transfer function in the first cycle; and
filter using second direct transfer function in the second cycle.

406. The data signal for fast decoding of claim 405, wherein
said second cycle is active during horizontal filtering of each second pixel; and
said first cycle is active during horizontal filtering of other pixels.
- 5 407. The data signal for fast decoding of claim 405, wherein
said second cycle is active during vertical filtering of each second line; and
said first cycle is active during vertical filtering of other lines.
- 10 408. The data signal for fast decoding of claim 405, wherein
said step that inverse non-stationary filters **further causes the machine to:**
multiply an input with a reciprocal value of a first gain number to produce a first result;
multiply an input with a reciprocal value of a second gain number to produce a second
result;
select the first result in each second cycle to produce an input sample for a plurality of steps
15 that successive inverse non-stationary cell filter; and
select the second result in each first cycle to produce an input sample for a plurality of steps
that successive inverse non-stationary cell filter.
- 20 409. The data signal for fast decoding of claim 405, wherein
said step that inverse non-stationary cell filters **further causes the machine to:**
delay an input sample for w samples to produce a plurality of even and odd indexed
delayed results in each first cycle;
multiply each even indexed delayed result with appropriate first filter coefficient selected
from a group of first filter coefficients to produce first results;
25 add together all first results to produce a third result;
add the third result with the input sample to produce a fifth result;
delay the fifth result for w samples to produce a plurality of even and odd indexed delayed
results in each second cycle;
multiply the input sample and each odd indexed delayed result with appropriate second
30 filter coefficient selected from a group of second filter coefficients to produce second results;
add together all second results to produce a fourth result;
add the fourth result with last delayed result to produce a sixth result;

output the sixth result in each first cycle; and
output the last delayed result in each second cycle.

- 5 **410.** The data signal for fast decoding of claim 409, wherein
at least one said step that multiplies causes the machine to
perform an operation selected from a group consisting of:
shift and bit remap.
- 10 **411.** The data signal for fast decoding of claim 409, wherein
said step that inverse non-stationary cell filters further causes the machine to
transform the third result by a first function N_1 to produce a seventh result;
transform the fourth result by a first function N_2 to produce an eight result;
add the seventh result with the input sample to produce a fifth result; and
add the eight result with last delayed result to produce a sixth result.
- 15 **412.** The data signal for fast decoding of claim 411, wherein
at least one said step that transforms causes the machine to
perform an operation selected from a group consisting of:
shift and bit remap.
- 20 **413.** The data signal for fast decoding of claim 405, wherein
said step that inverse non-stationary cell filters further causes the machine to:
delay an input sample for w samples to produce a delayed result in each first cycle;
multiply the delayed result with a second filter coefficient to produce a second result;
25 add the second result with the input sample to produce a fourth result;
delay the fourth result for w samples to produce the delayed result in each second cycle;
multiply the input sample with a first filter coefficient to produce a first result;
add the first result with the delayed result to produce a third result;
output the third result in each first cycle; and
30 output the delayed result in each second cycle.

414. The data signal for fast decoding of claim 413, wherein
at least one said step that multiplies **causes the machine to**
perform an operation selected from a group consisting of:
shift and bit remap.
- 5
415. The data signal for fast decoding of claim 414, wherein
said operation **causes the machine to**
shift right for one bit position.
- 10 416. The data signal for fast decoding of claim 414, wherein
said operation **causes the machine to**
shift right data for two bit positions.
417. The data signal for fast decoding of claim 416, wherein
15 said step that adds the second result **causes the machine to** subtract the second result from
the input sample to produce a fourth result; and
said step that adds the first result **causes the machine to** subtract the first result from the
delayed result to produce a third result.
- 20 418. The data signal for fast decoding of claim 405, wherein
said step that inverse non-stationary cell filters **further causes the machine to:**
delay an input sample for w samples to produce a first delayed result in each first cycle;
delay the first delayed result for w samples to produce a second delayed result;
delay the second delayed result for w samples to produce a third delayed result;
25 multiply the input sample with a first filter coefficient to produce a first result;
multiply the first delayed result with a second filter coefficient to produce a second result;
multiply the second delayed result with a third filter coefficient to produce a third result;
multiply the third delayed result with a fourth filter coefficient to produce a fourth result;
add the fourth result with the second result to produce a sixth result;
30 add the first result with the third result to produce a fifth result;
add the sixth result with the input sample to produce an eight result;
delay the eight result for w samples to produce the first delayed result in each second cycle;

add the fifth result with the third delayed result to produce a seventh result;
output the seventh result in each first cycle; and
output the third delayed result in each second cycle.

- 5 **419.** The data signal for fast decoding of claim 418, wherein
at least one said step that multiplies **causes the machine to**
perform an operation selected from a group consisting of:
shift and bit remap.
- 10 **420.** The data signal for fast decoding of claim 419, wherein
said operation **causes the machine to**
shift right data for four bit positions.
- 15 **421.** The data signal for fast decoding of claim 420, wherein
said step that adds the fourth result **causes the machine to** subtract the fourth result from
the second result to produce a sixth result; and
said step that adds the first result **causes the machine to** subtract the first result from the
third result to produce a fifth result.
- 20 **422.** The data signal for fast decoding of claim 392, wherein
said step that estimates the probabilities of symbols within the specified contexts **causes**
the machine to
update adaptive histograms.
- 25 **423.** The data signal for fast decoding of claim 422, wherein
said step that updates adaptive histograms **causes the machine to**
low-pass filter probabilities selected from a group consisting of:
probabilities of occurrences of a current symbol x ; and
cumulative probabilities of occurrences of all symbols preceding said current symbol x .

424. The data signal for fast decoding of claim 423, wherein
said step that updates adaptive histograms **further causes the machine to**
adapt a dominant pole during said low-pass filtering.
- 5 425. The data signal for fast decoding of claim 424, wherein
said step that adapts a dominant pole **causes the machine to**
halve value of the dominant pole in each adaptation cycle.
- 10 426. The data signal for fast decoding of claim 392, wherein
said step that entropy decodes is a step that range decode, **causing the machine to**
multiply a prescaled range r with a number $Q(x)$ selected from a group consisting of:
a number $U(x)$ of occurrences of all symbols preceding a current symbol x , to produce a
range correction $t = r \cdot U(x)$; and
a number $u(x)$ of occurrences of the current symbol x , to produce a range $R = r \cdot u(x)$.
- 15 427. The data signal for fast decoding of claim 426, wherein
said step that multiplies **causes the machine to:**
simplified multiply a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ with said number $Q(x)$ to produce a first
result; and
20 shift left the first result for l bit positions.
428. The data signal for fast decoding of claim 426, wherein
said step that multiplies **causes the machine to**
shift left said number $Q(x)$ for l bit positions.
- 25 429. The data signal for fast decoding of claim 426, wherein
said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shift left said number $Q(x)$ for one bit position to produce a first result, when said small
30 number V is any odd number higher or equal 3;

add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.

430. The data signal for fast decoding of claim 426, wherein
5 said step that multiplies **causes the machine to:**
zero a first result, when a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ is equal 1;
shift left said number $Q(x)$ for one bit position to produce a first result, when said small
number V is equal 3;
shift left said number $Q(x)$ for two bit positions to produce a first result, when said small
10 number V is any odd number higher or equal 5;
add said number $Q(x)$ with the first result to produce a second result; and
shift left the second result for l bit positions.

431. The data signal for fast decoding of claim 392, wherein
15 said step that entropy decodes is a step that range decode, **causing the machine to**
divide a range R with a number $Total$ of occurrences of all symbols, to produce a prescaled
range $r = \lfloor R/Total \rfloor$.

432. The data signal for fast decoding of claim 431, wherein
20 said step that divides **causes the machine to**
shift right said range R for $w_3 = \log_2(Total)$ bit positions.

433. The data signal for fast decoding of claim 392, wherein
said step that entropy decodes is a step that range decode, **causing the machine to**
25 divide a bottom range limit B with a prescaled range r , to produce a range correction
 $t = \lfloor B/r \rfloor$.

434. The data signal for fast decoding of claim 433, wherein
said step that divides **causes the machine to:**

simplified divide said bottom range limit B with a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a first result; and

shift right the first result for l bit positions.

5 **435.** The data signal for fast decoding of claim 433, wherein

said step that divides **causes the machine to:**

multiply said bottom range limit B with a first predefined number, dependent on a small number $V = \lfloor r \cdot 2^{-l} \rfloor$ to produce a first result; and

shift right the first result for a sum of l and a second predefined number of bit positions,
10 dependent on said small number V .

436. The data signal for fast decoding of claim 392, wherein

said step that estimates the probabilities of symbols within the specified contexts **causes the machine to**

15 reconstruct transformation coefficient C , using a magnitude-set index MS , a sign S and a residual R .

437. The data signal for fast decoding of claim 436, wherein

said step that entropy decodes **causes the machine to**

20 decode said residual R using inverse variable length coding (INVLC).

438. The data signal for fast decoding of claim 436, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

25 context model a transformation coefficient to be decoded, using already decoded transformation coefficients.

439. The data signal for fast decoding of claim 438, wherein

said already decoded transformation coefficients are located north-east, north, north-west
30 and west from the transformation coefficient to be decoded.

440. The data signal for fast decoding of claim 438, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

determine a mean value \overline{MS} as the mean value of magnitude-set indexes MS_i of said
5 already decoded transformation coefficients.

441. The data signal for fast decoding of claim 440, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

10 limit a maximum mean value \overline{MS} by a constant ML to produce a magnitude context MC .

442. The data signal for fast decoding of claim 441, wherein

said step that entropy decodes **causes the machine to**

range decode said magnitude-set index MS as a current symbol x , using said adaptive
15 magnitude histogram $h[MC]$.

443. The fast decoding of claim 442, wherein

said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**

20 update an adaptive magnitude histogram $h[MC]$ using said magnitude-set index MS as a current symbol x .

444. The data signal for fast decoding of claim 438, wherein

said step that estimates the probabilities of symbols within the specified contexts **further**
25 **causes the machine to**

determine a ternary context TC as the ternary code of sign values S_i of said already decoded transformation coefficients.

445. The data signal for fast decoding of claim 444, wherein
said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**
translate said ternary context *TC* into a sign context *SC* using CTX table.
- 5
446. The data signal for fast decoding of claim 445, wherein
said step that entropy decodes **causes the machine to**
range decode sign *S* as a current symbol *x*, using an adaptive sign histogram $g[SC]$.
- 10
447. The data signal for fast decoding of claim 446, wherein
said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**
update an adaptive sign histogram $g[SC]$ using decoded sign *S* as a current symbol *x*.
- 15
448. The data signal for fast decoding of claim 446, wherein
said step that estimates the probabilities of symbols within the specified contexts **further causes the machine to**
invert less probable decoded sign *S* using NEG table.
- 20
449. An article of manufacture for fast encoding of input data into output compressed data **comprising** a storage medium with a machine readable code **causing the machine to perform** the method of claim 113.
- 25
450. An article of manufacture for fast decoding of input compressed data into output data **comprising** a storage medium with a machine readable code **causing the machine to perform** the method of claim 168.
- 30
451. A data signal for fast encoding of input data into output compressed data embodied in a carrier wave **comprising** a machine readable code **causing the machine to perform** the method of claim 113.

452. A data signal for fast decoding of input compressed data into output data embodied in a carrier wave **comprising** a machine readable code **causing the machine to** perform the method of claim 168.

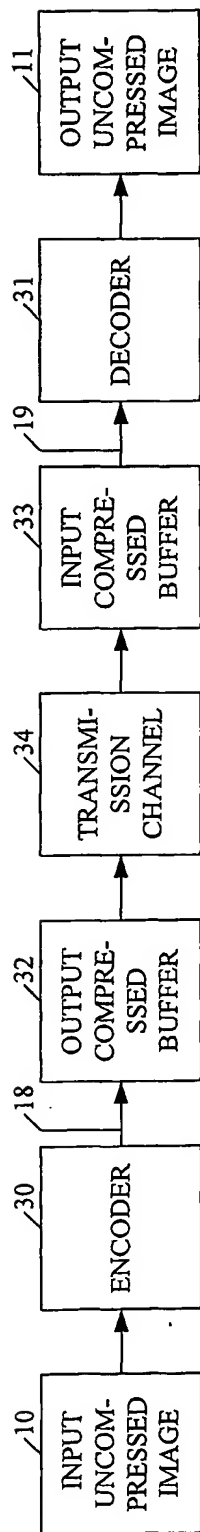


FIG. 1

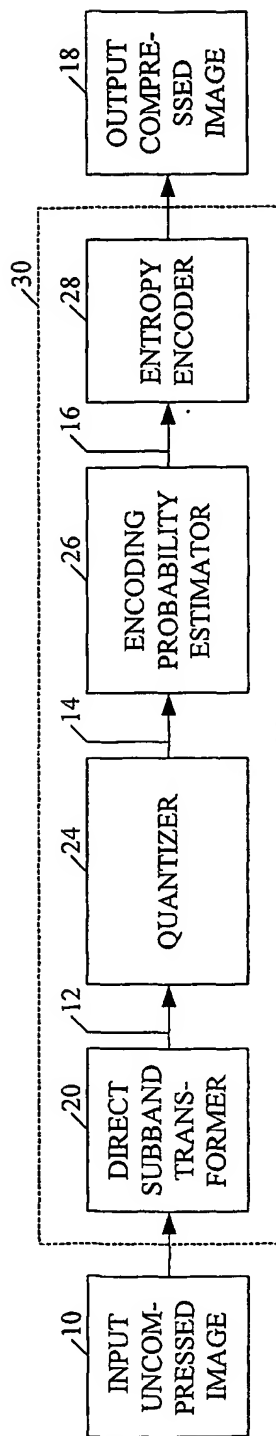


FIG. 2

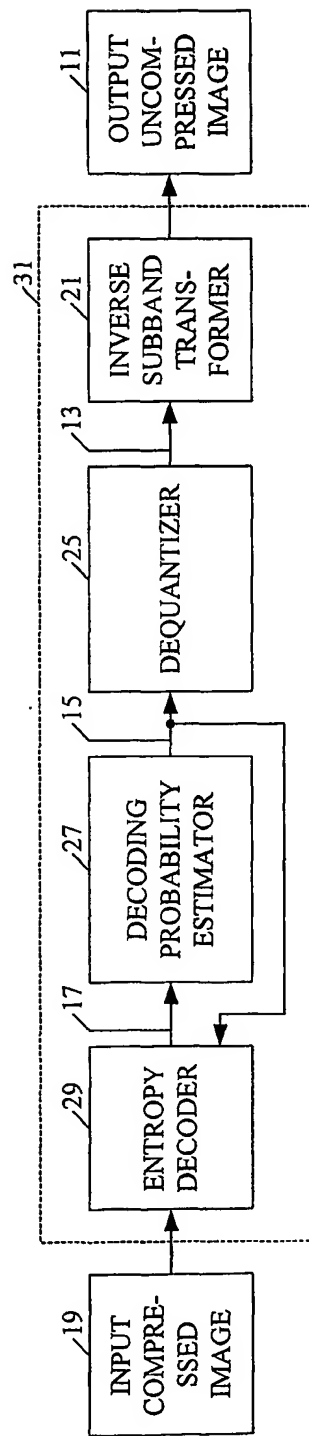


FIG. 3

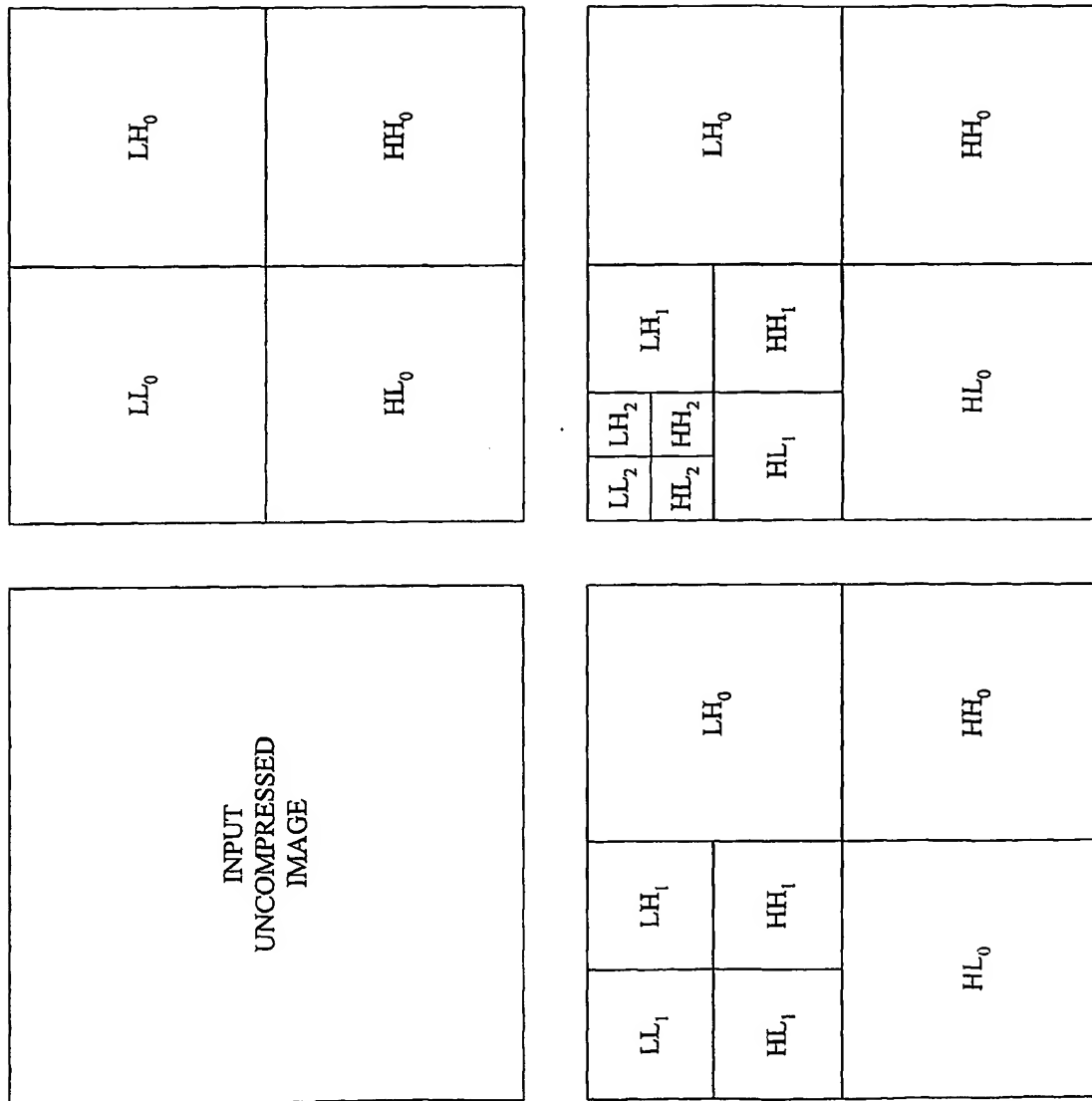


FIG. 4

3/34

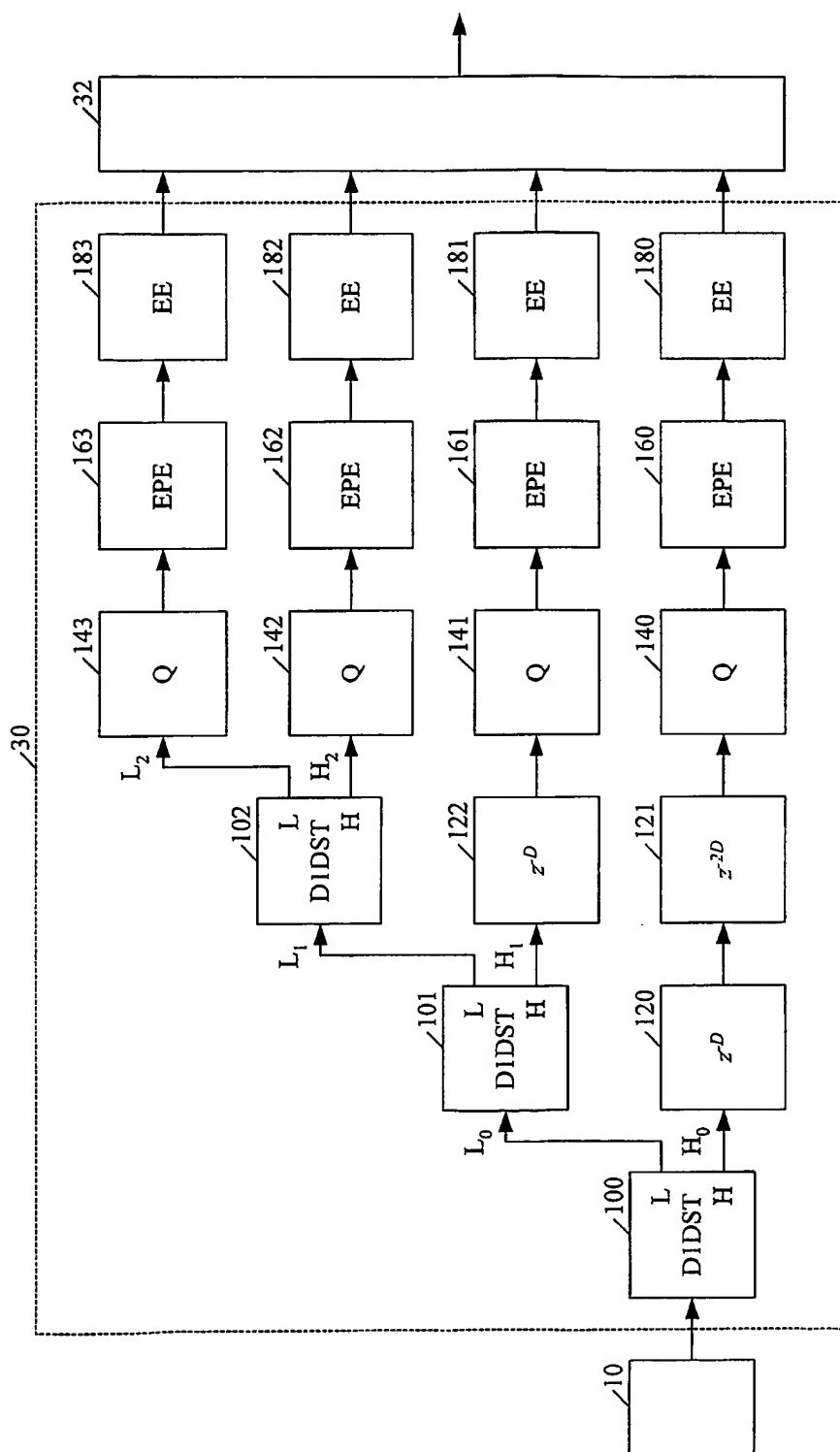


FIG. 5

4/34

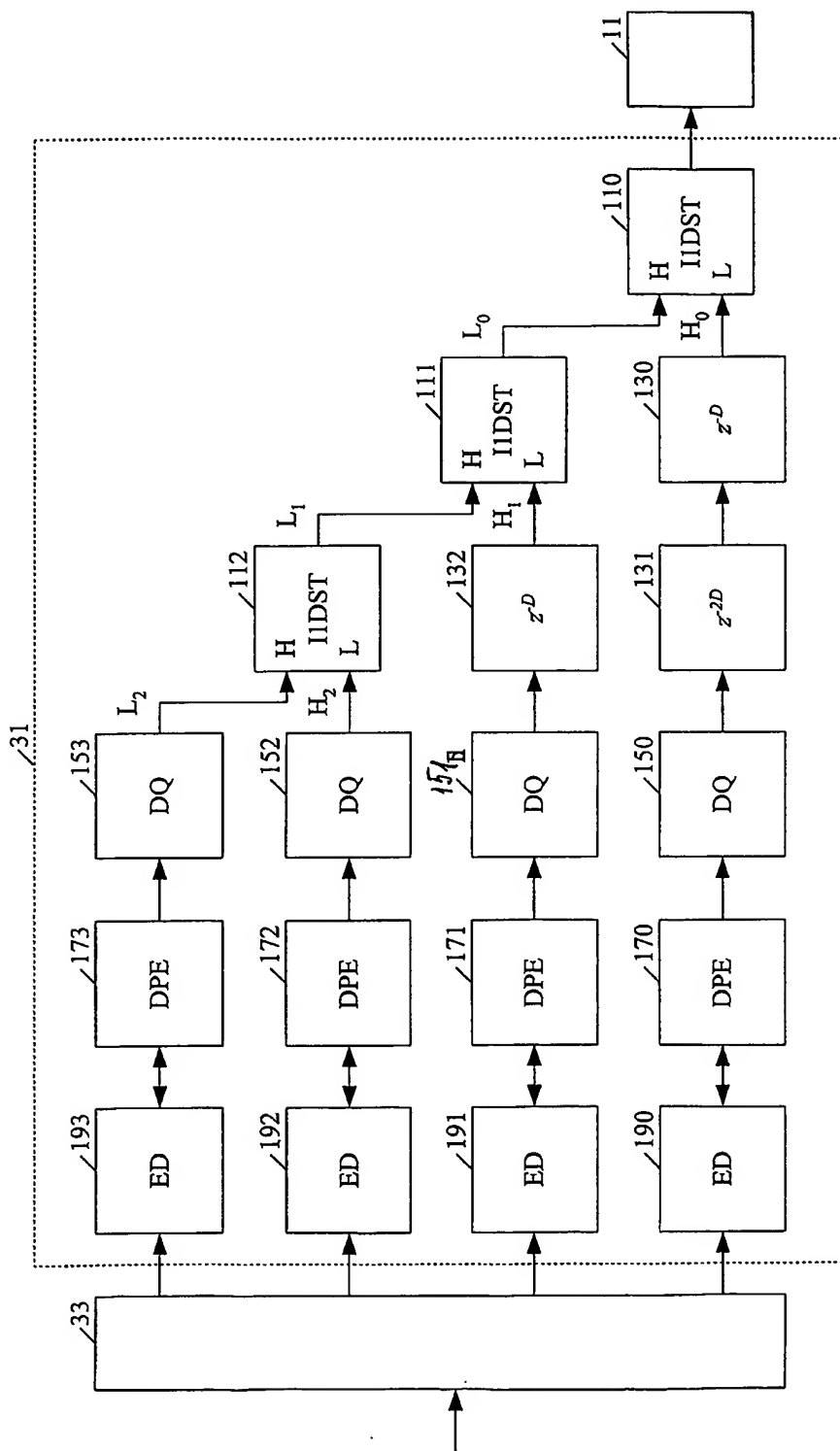
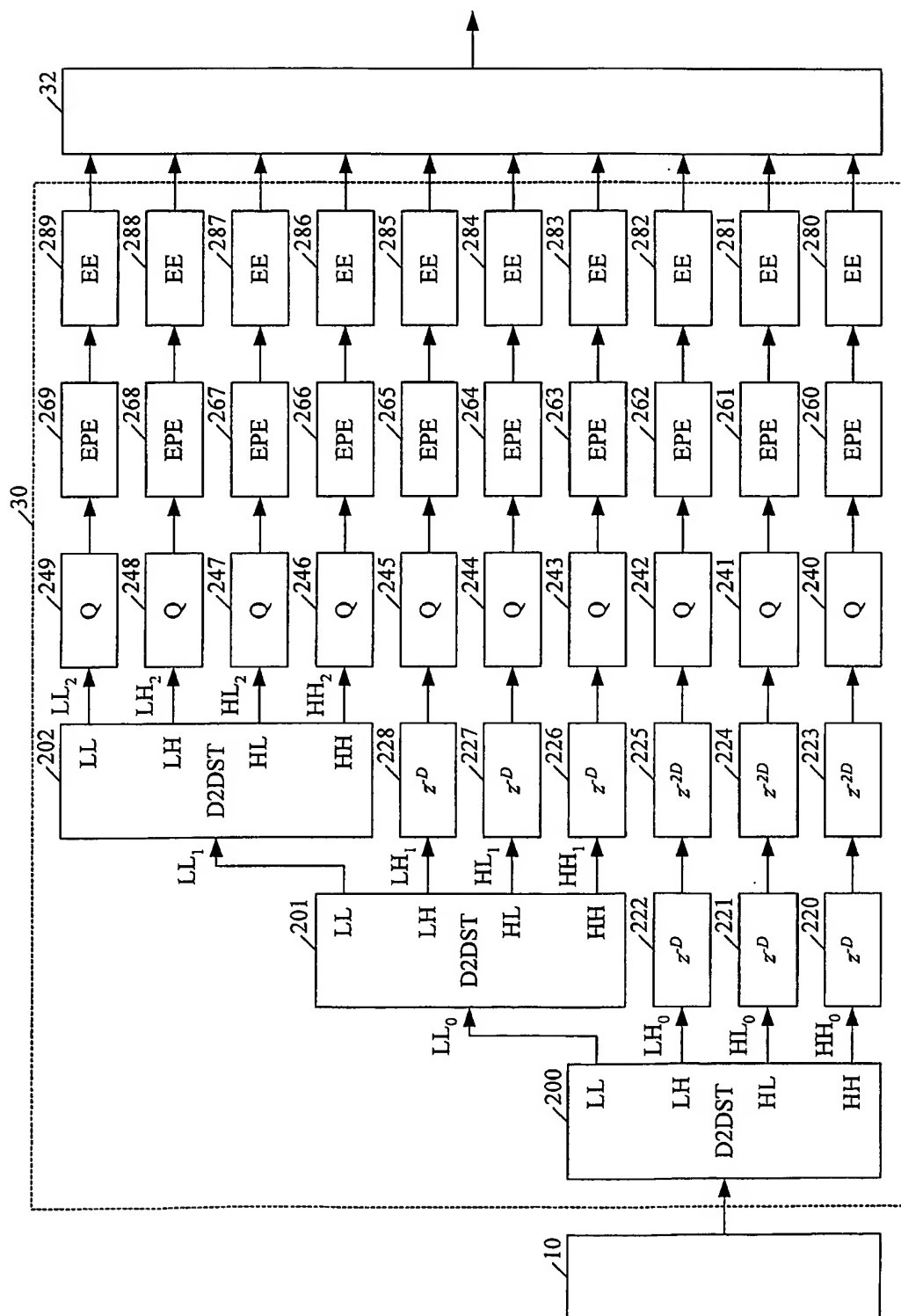


FIG. 6

5/34



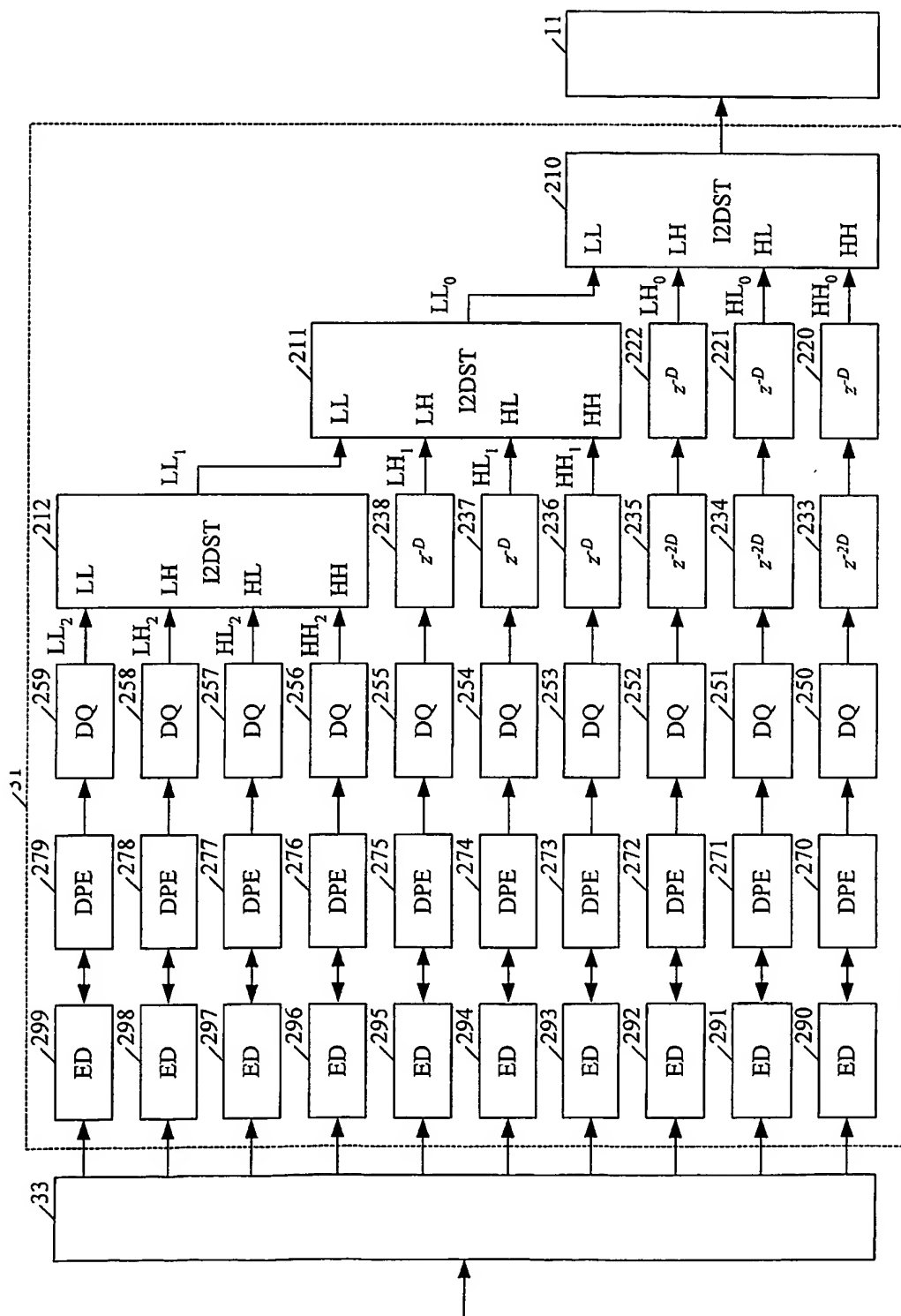


FIG. 8

7/34

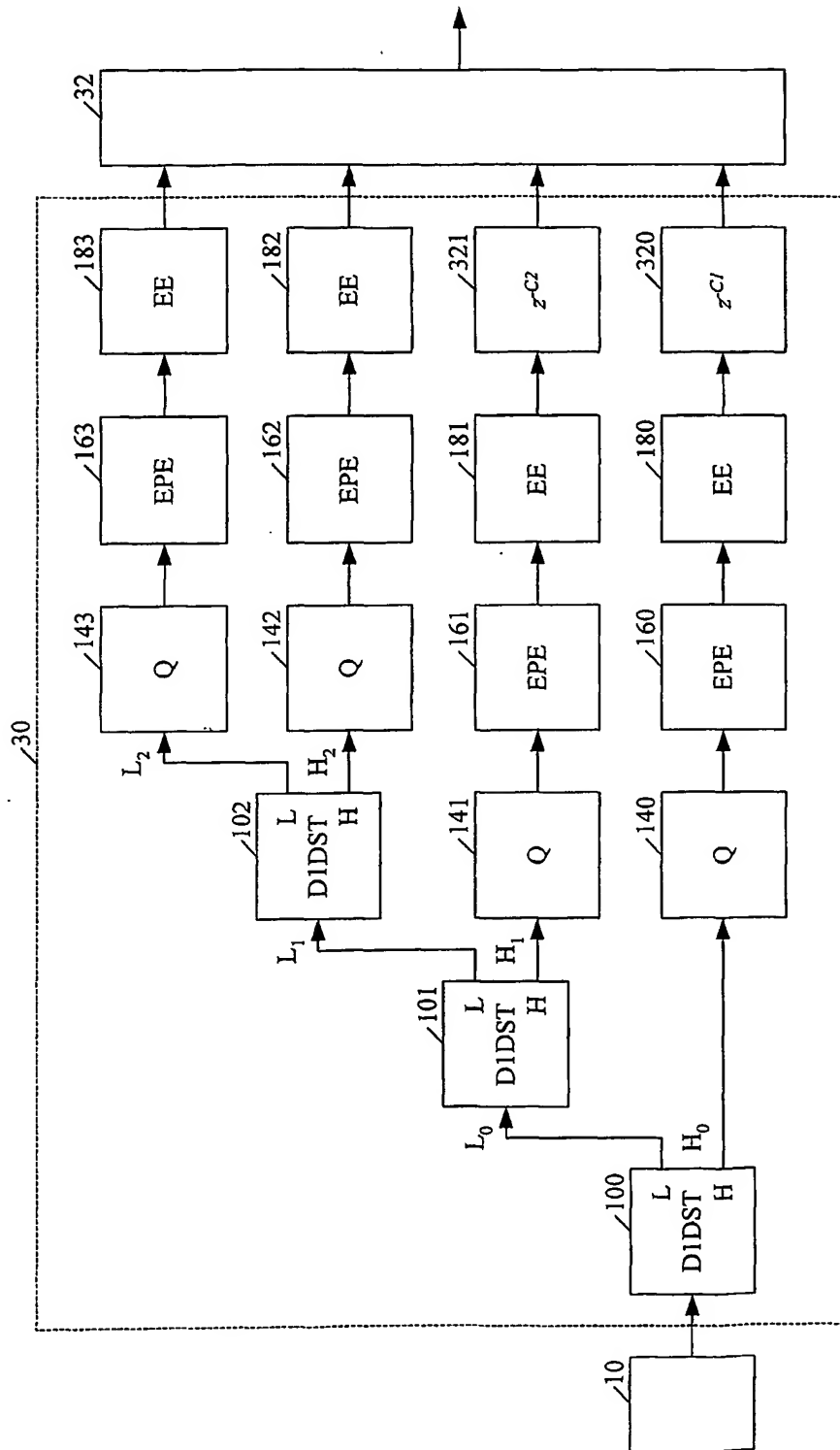


FIG. 9

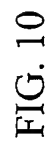


FIG. 10

9/34

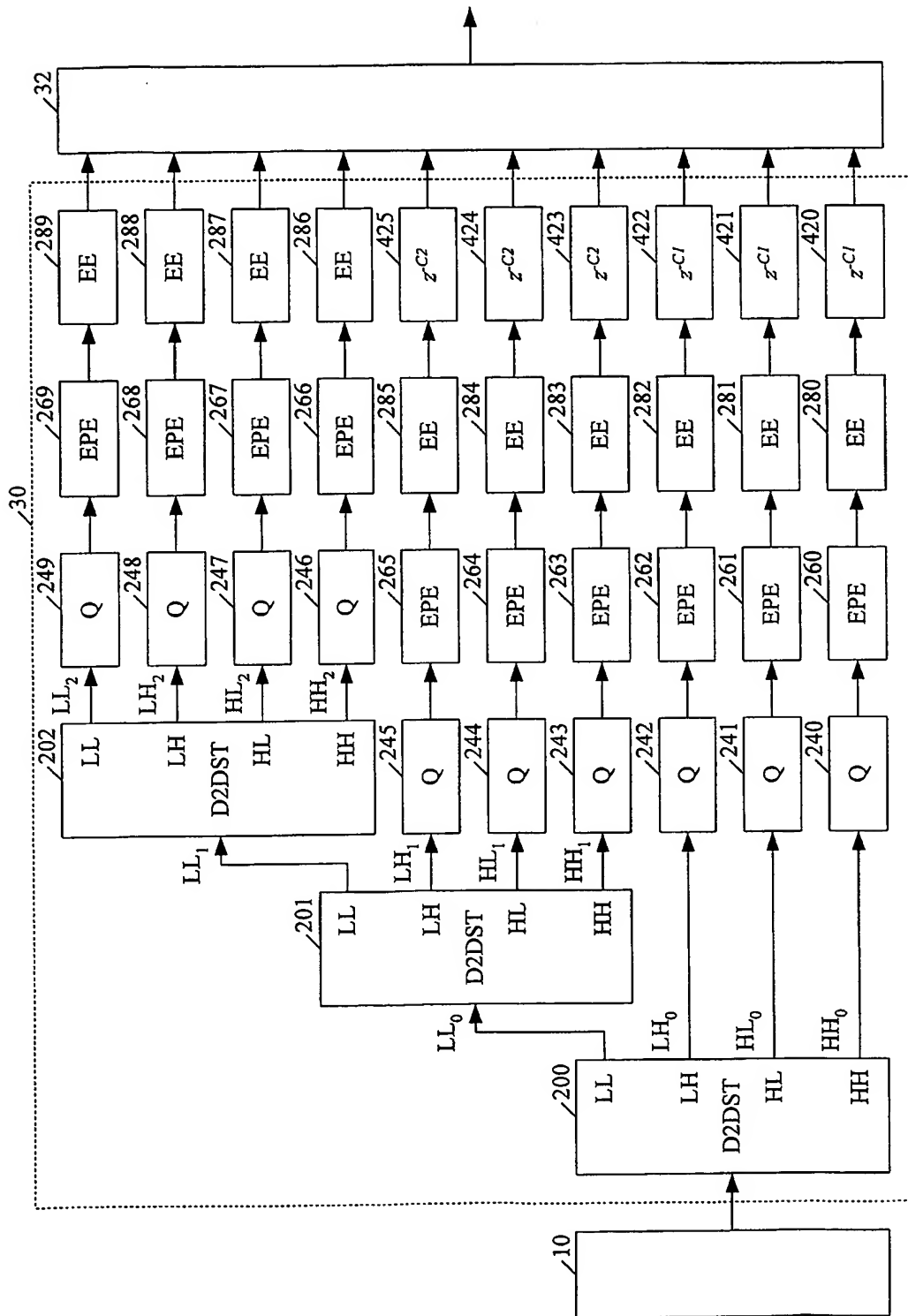


FIG. 11

10/34

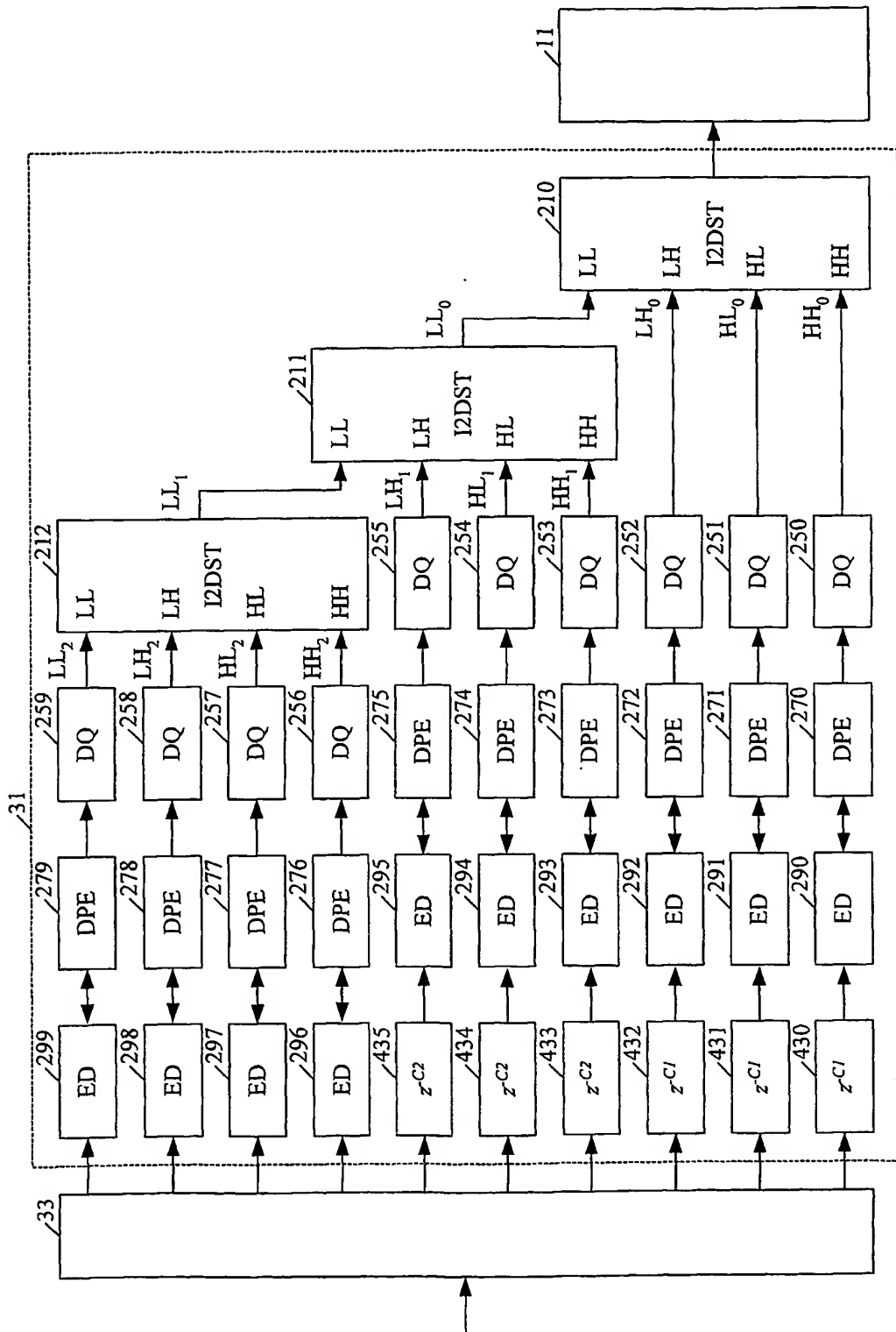


FIG. 12

11/34

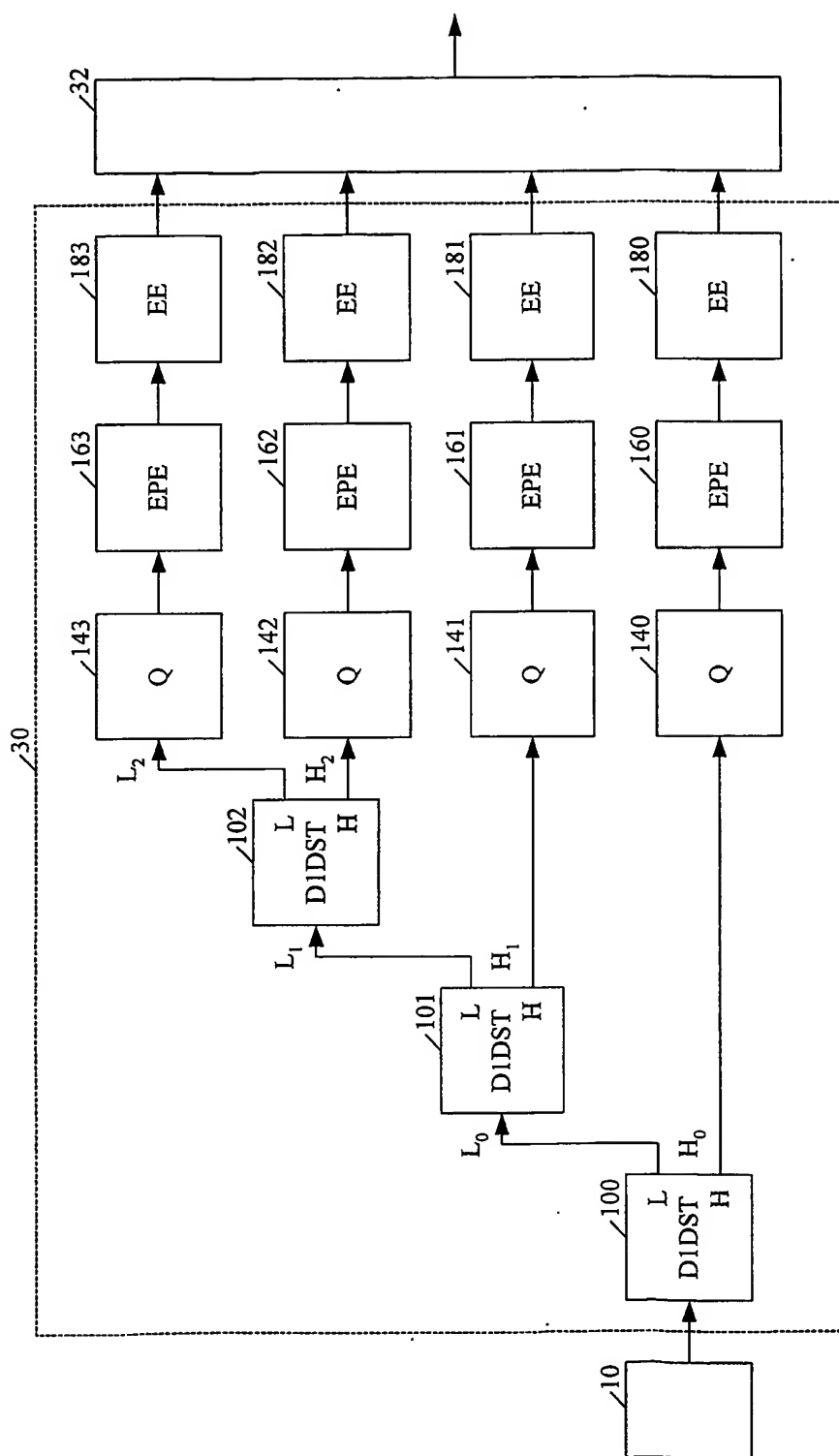


FIG. 13

12/34

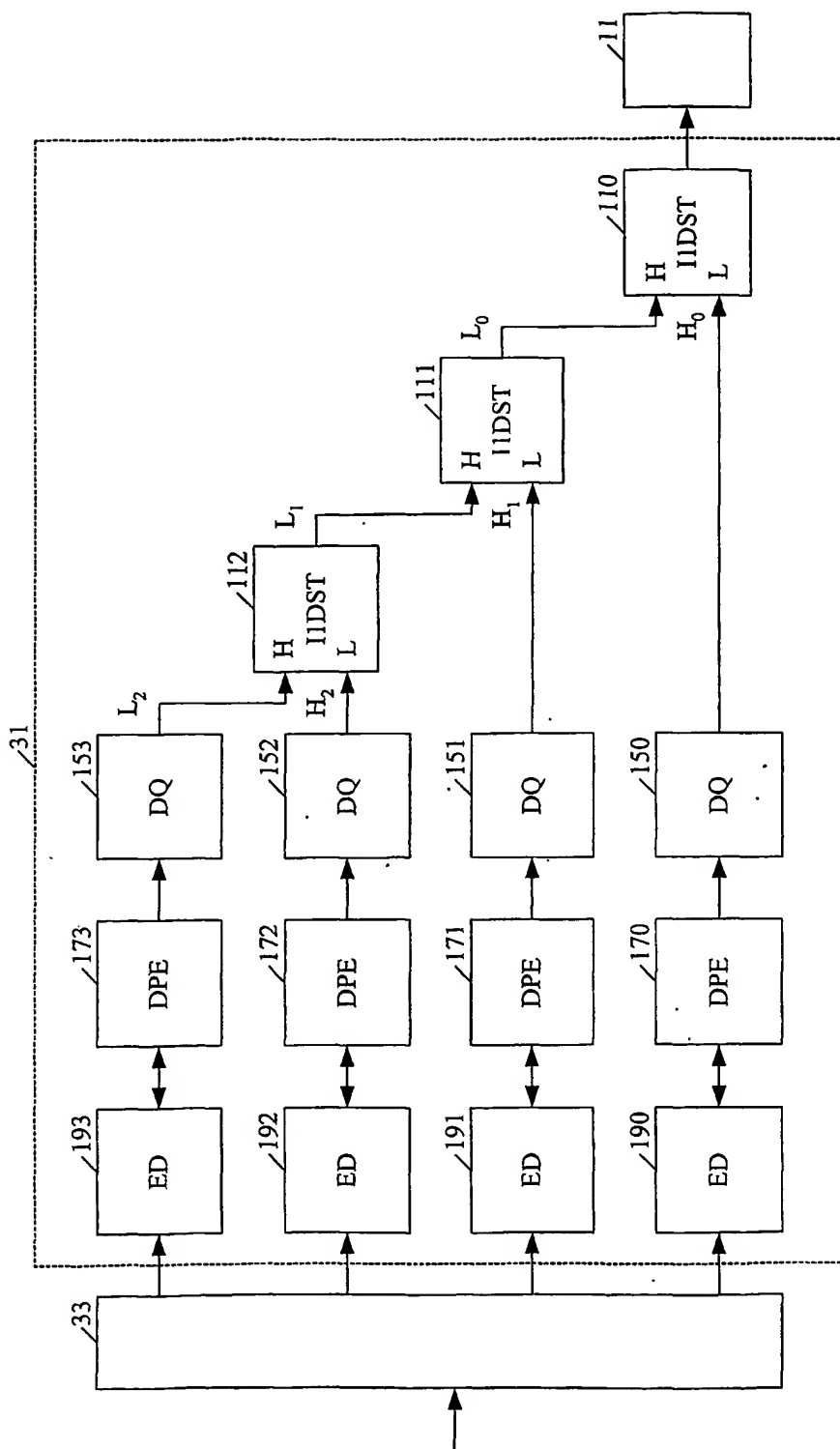


FIG. 14

13/34

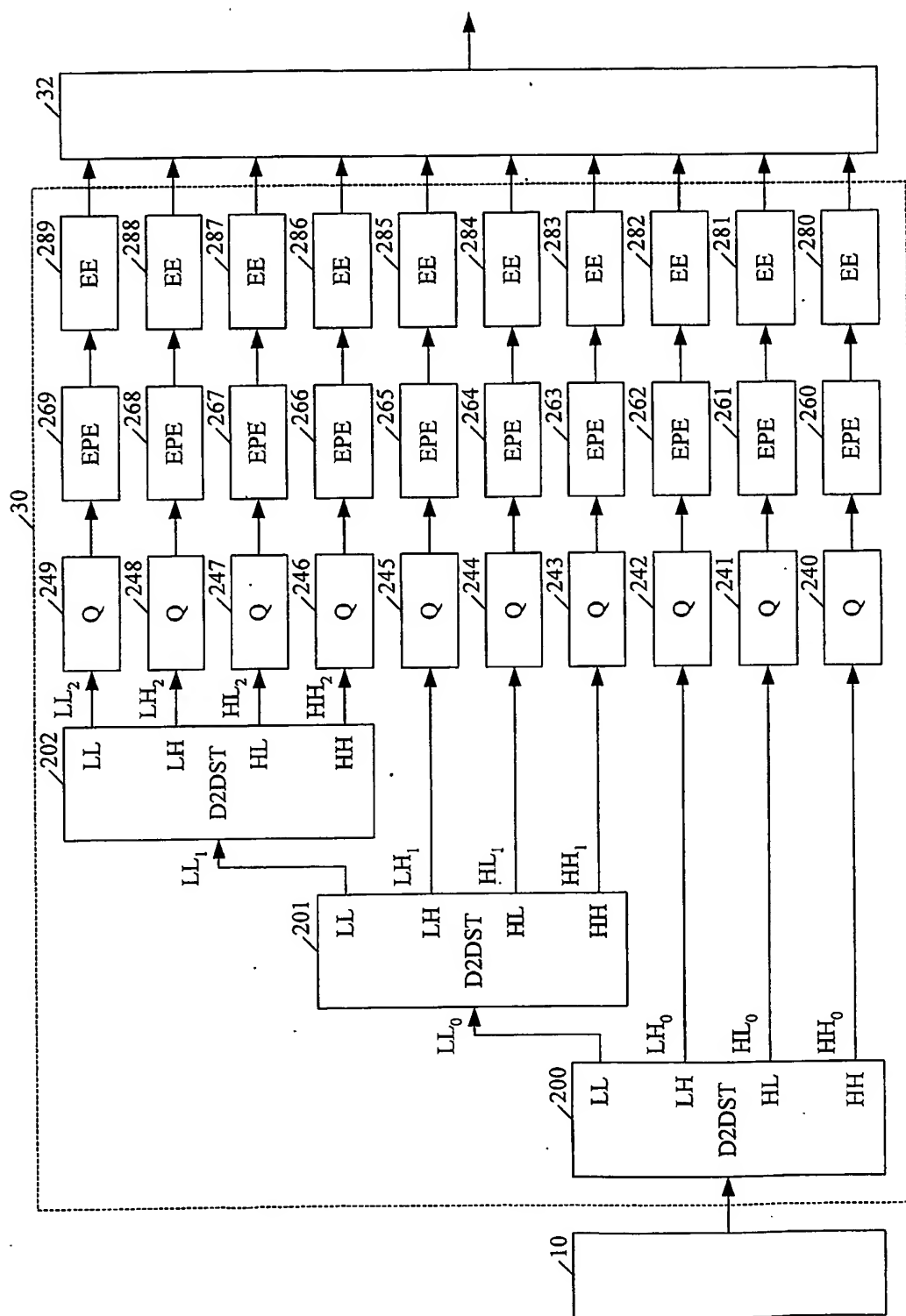


FIG. 15



FIG. 16

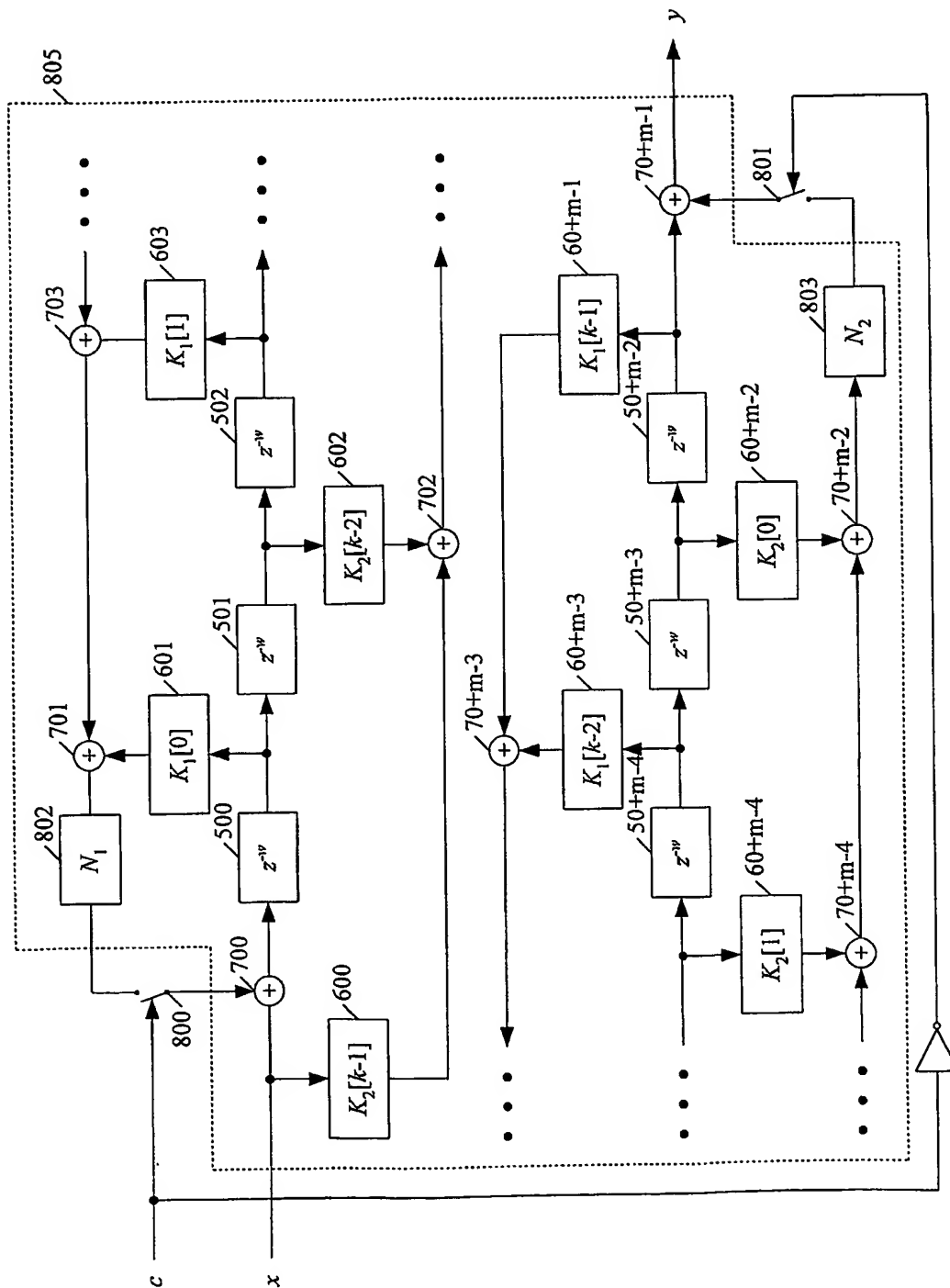
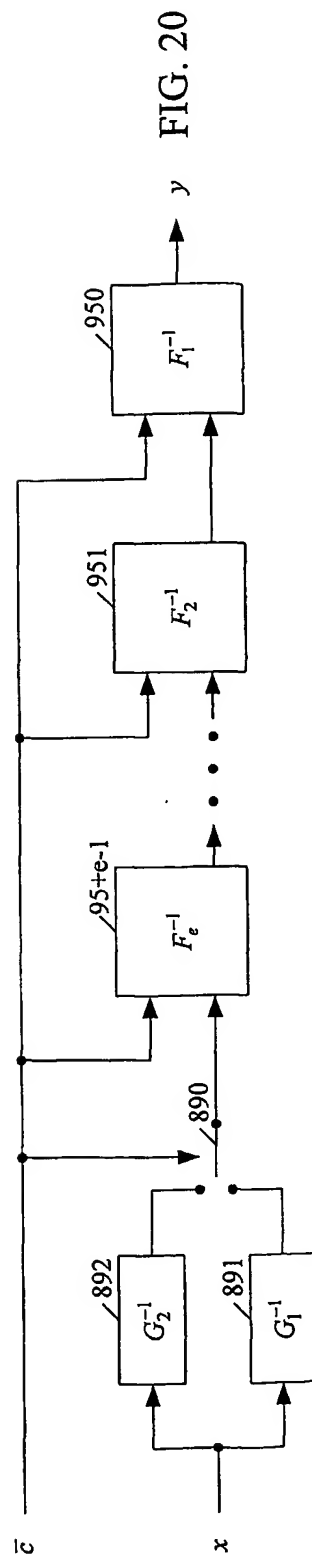
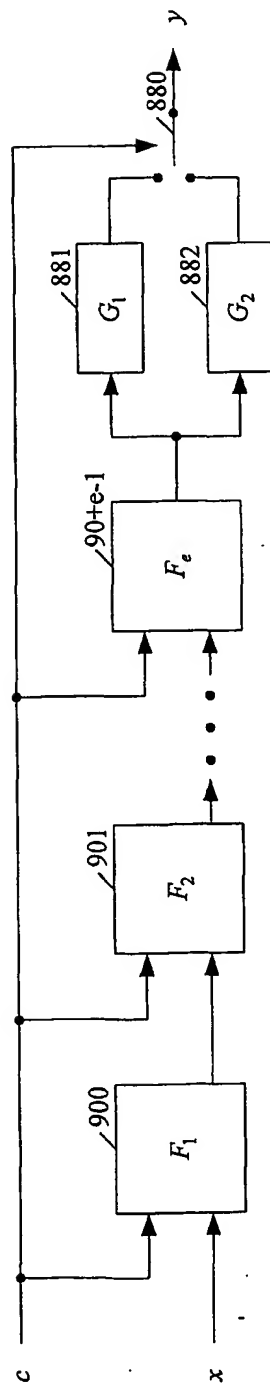
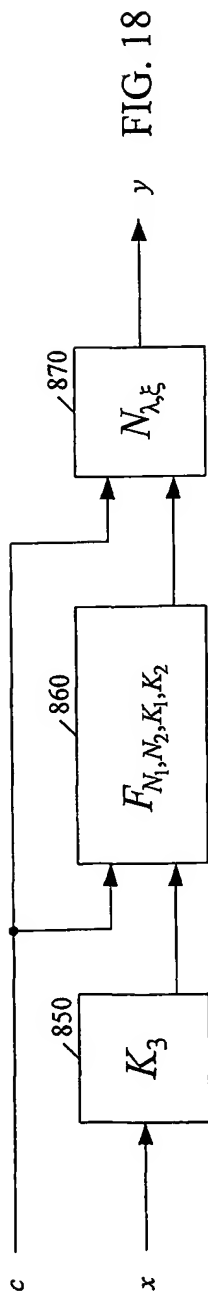


FIG. 17

16/34



17/34

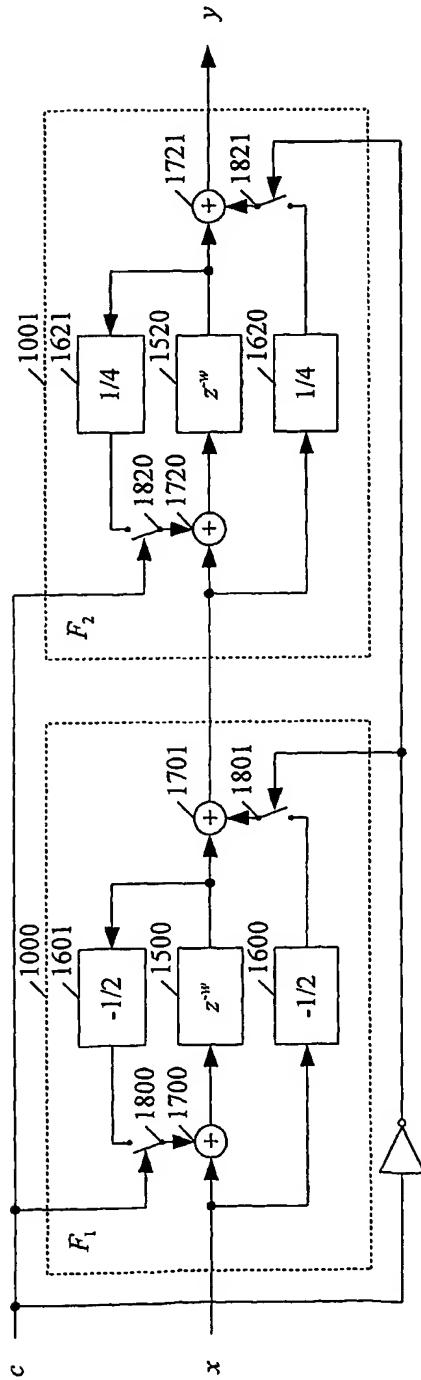


FIG. 21

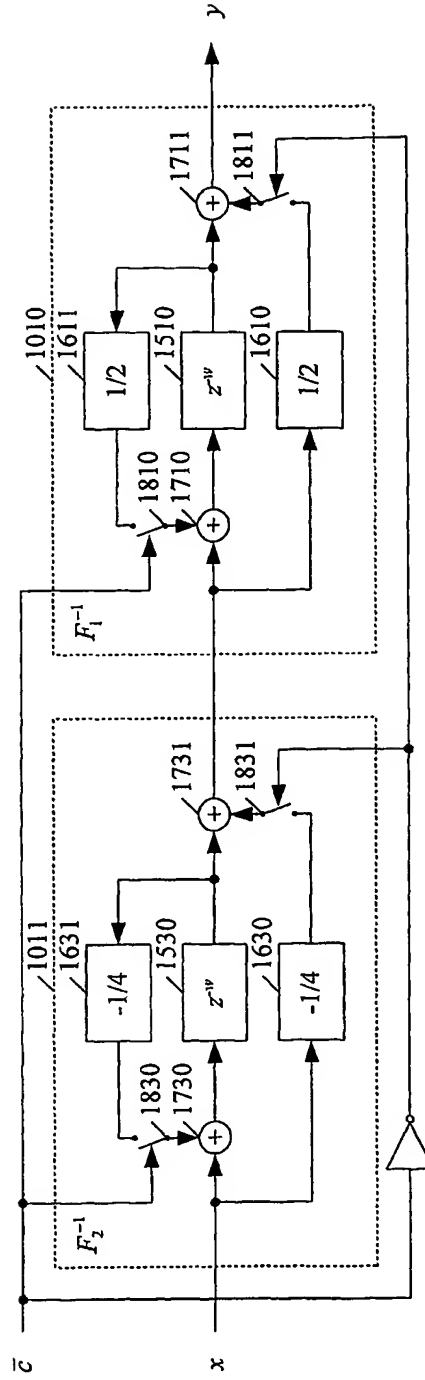


FIG. 22

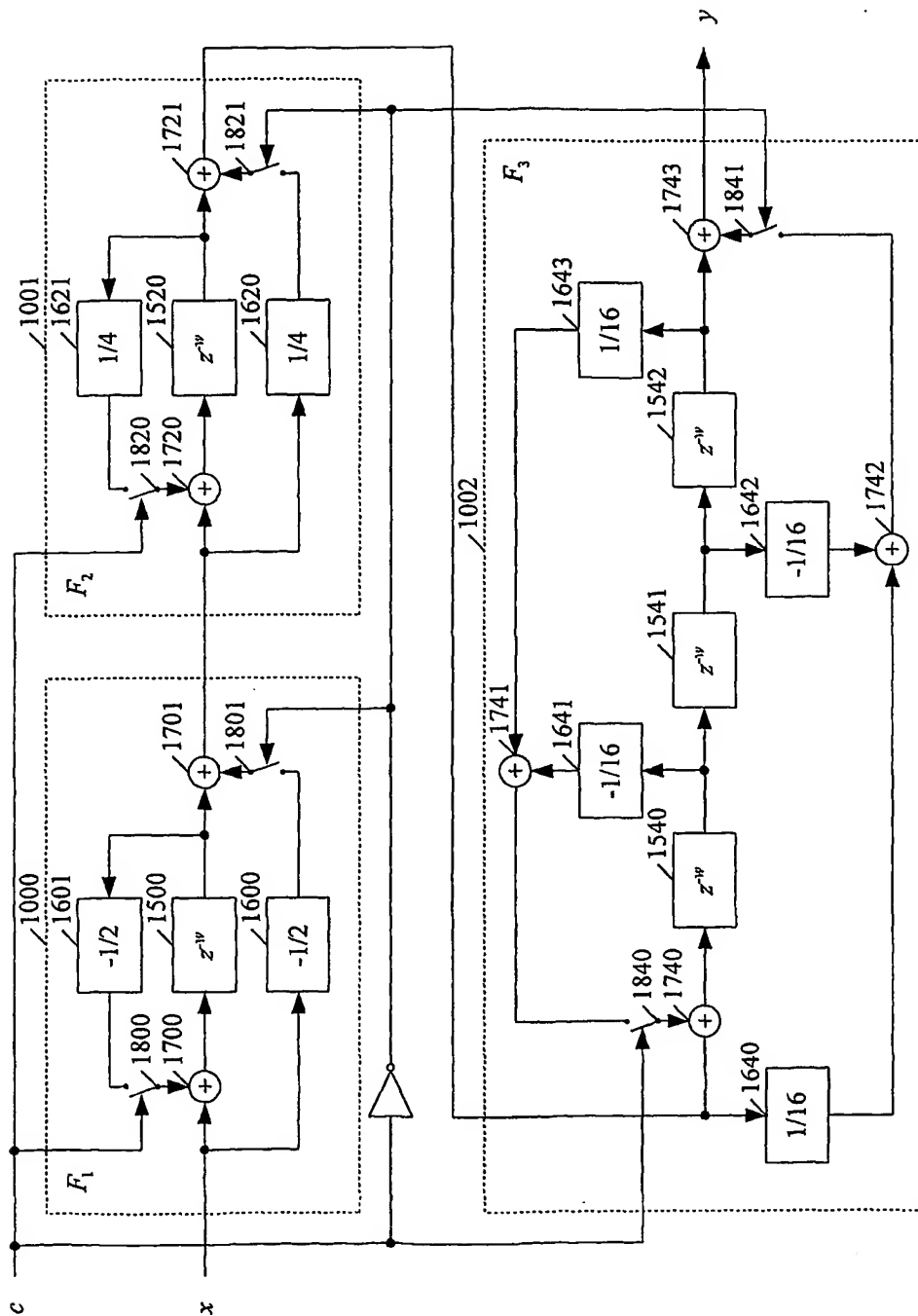


FIG. 23

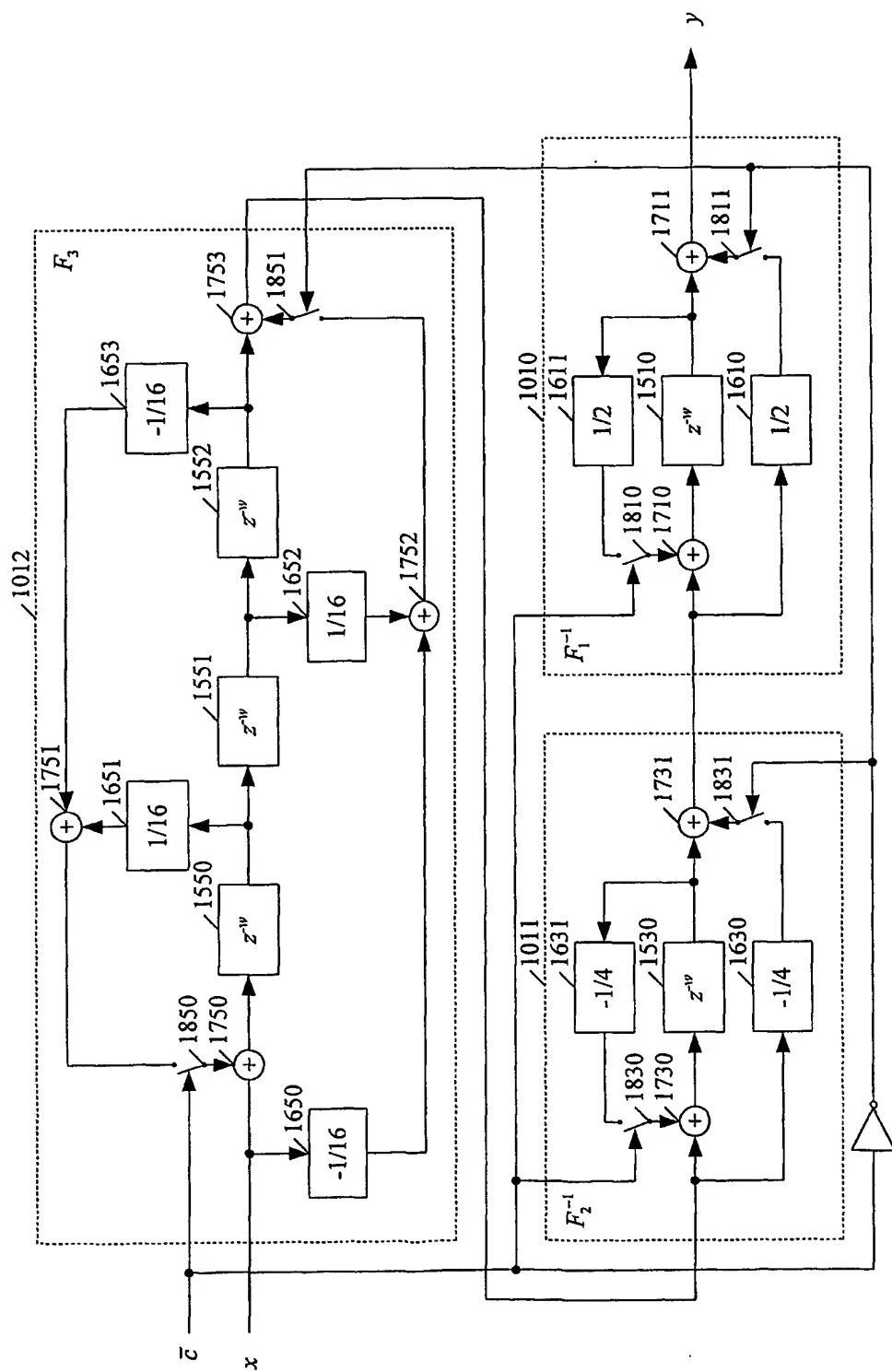


FIG. 24

20/34

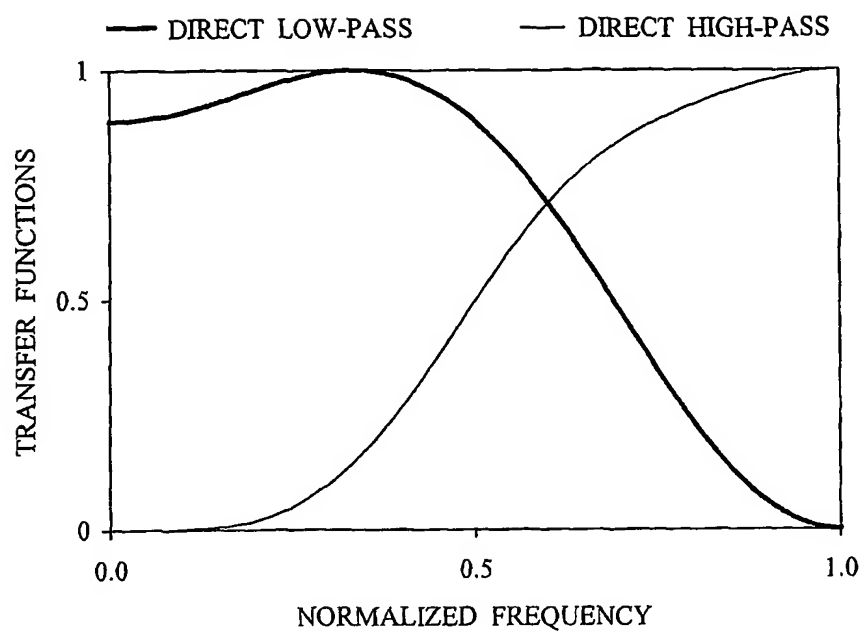


FIG. 25

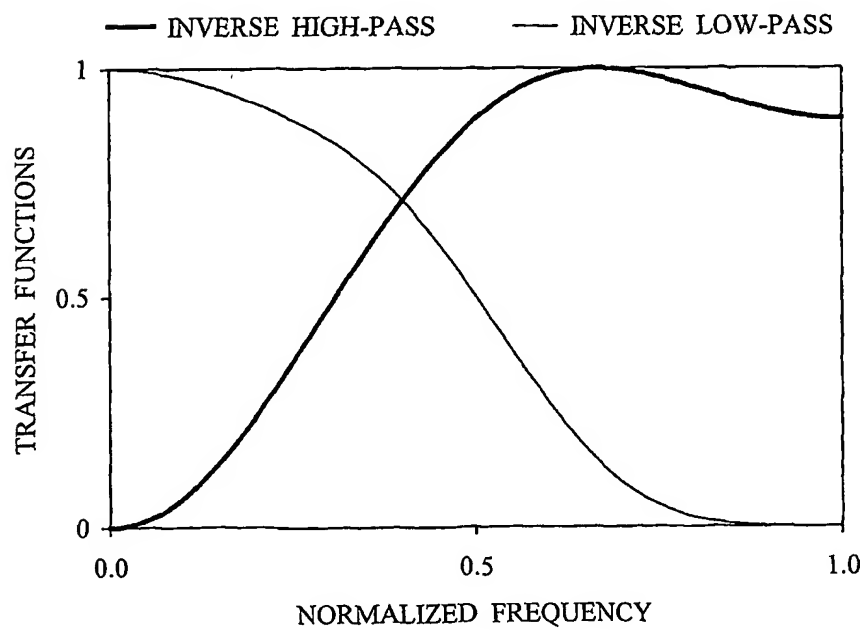


FIG. 26

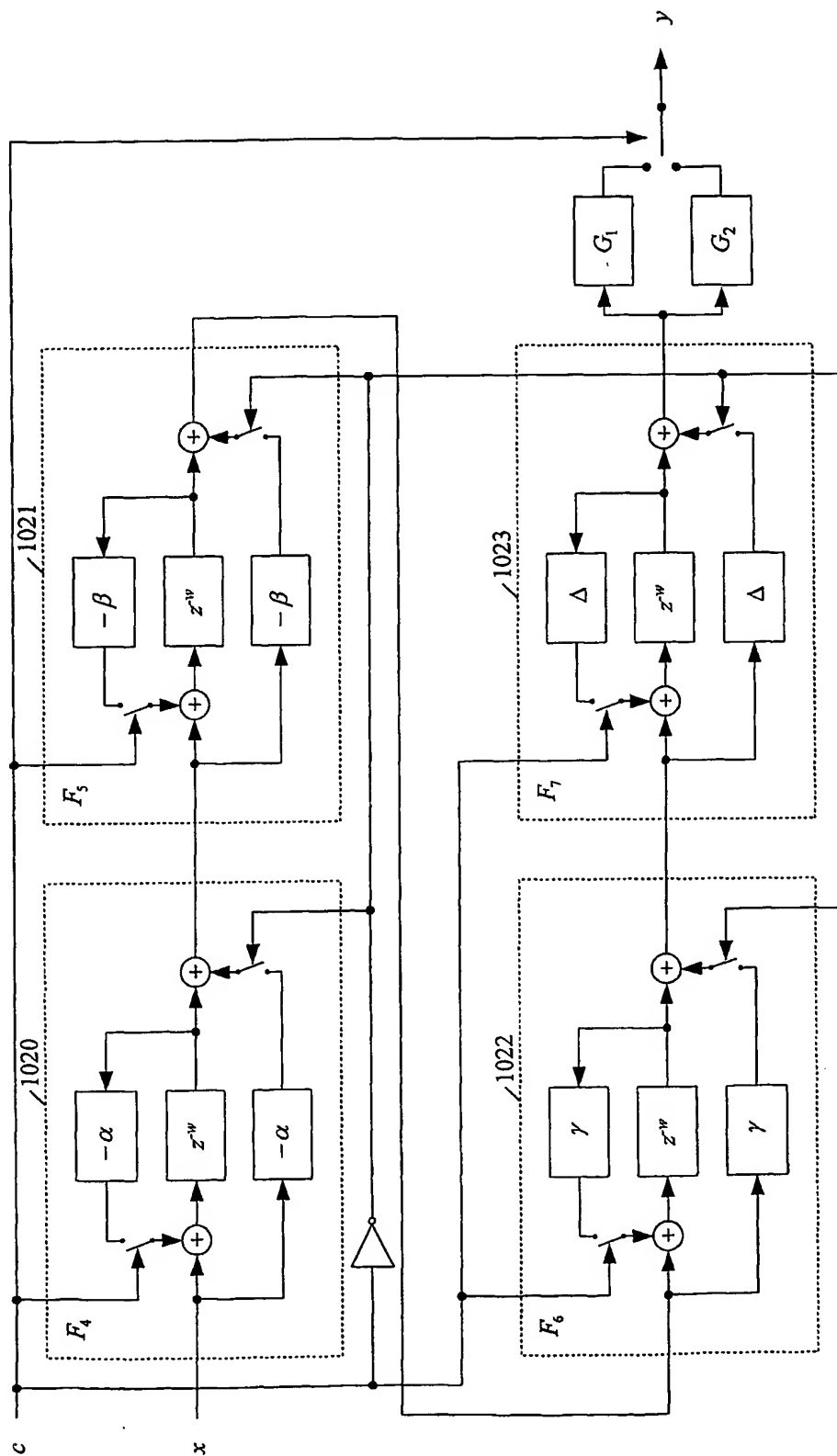


FIG. 27

22/34

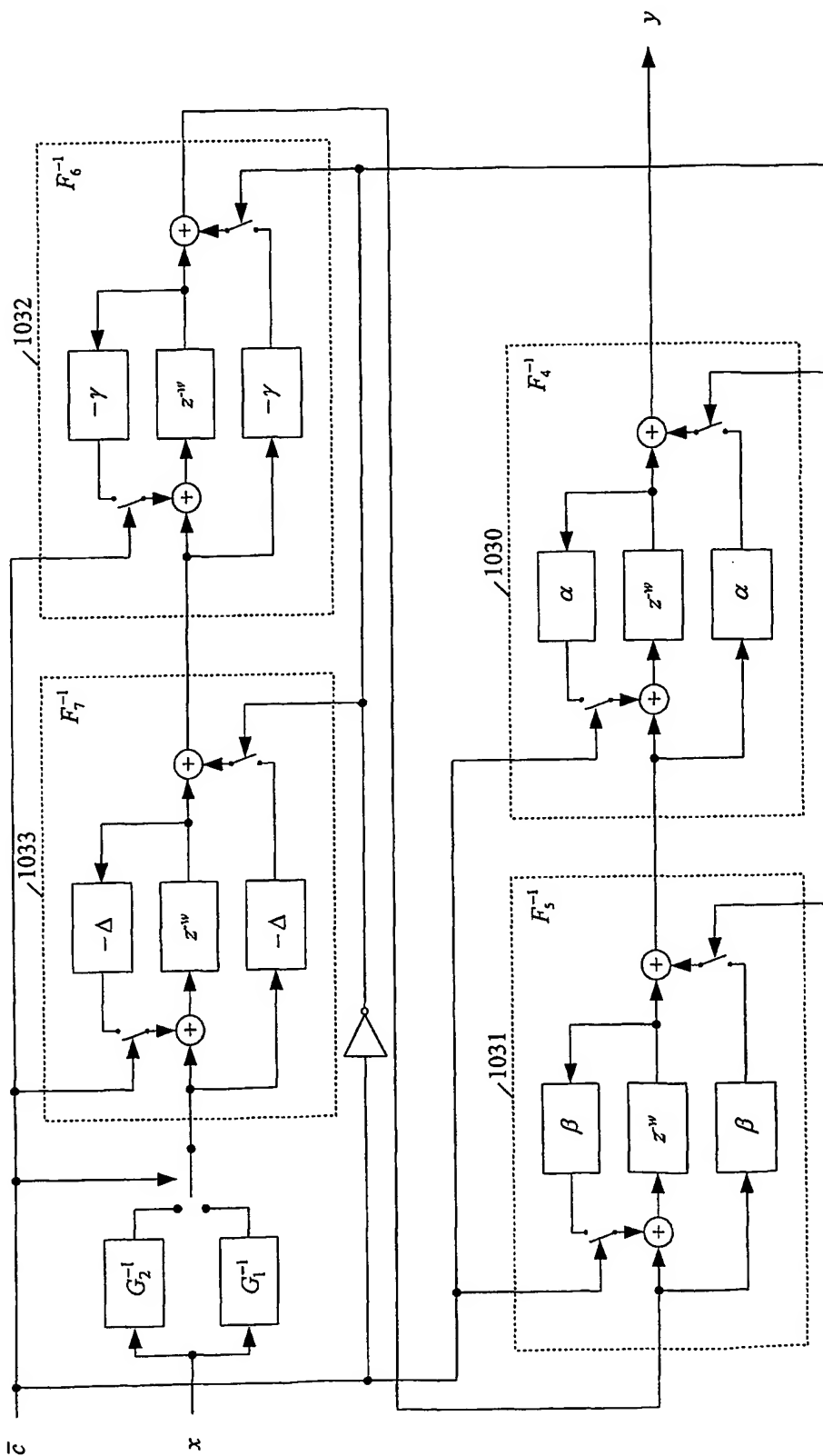


FIG. 28

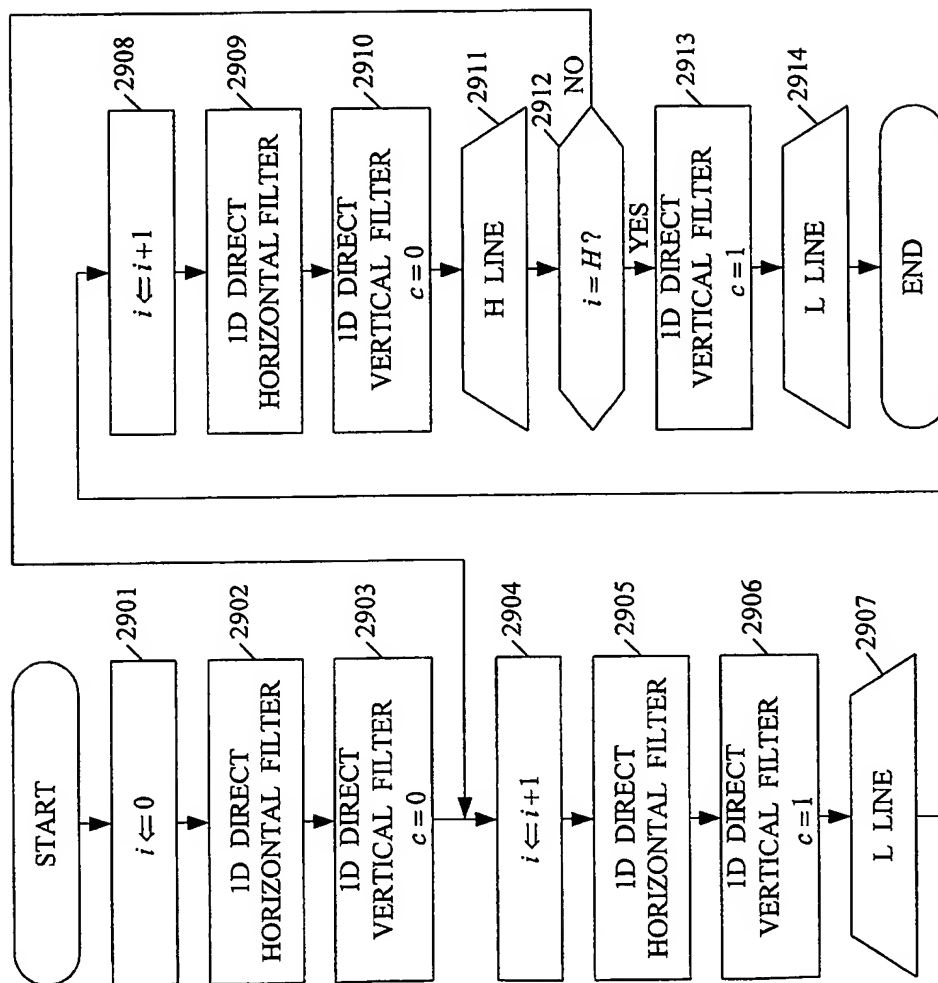


FIG. 29

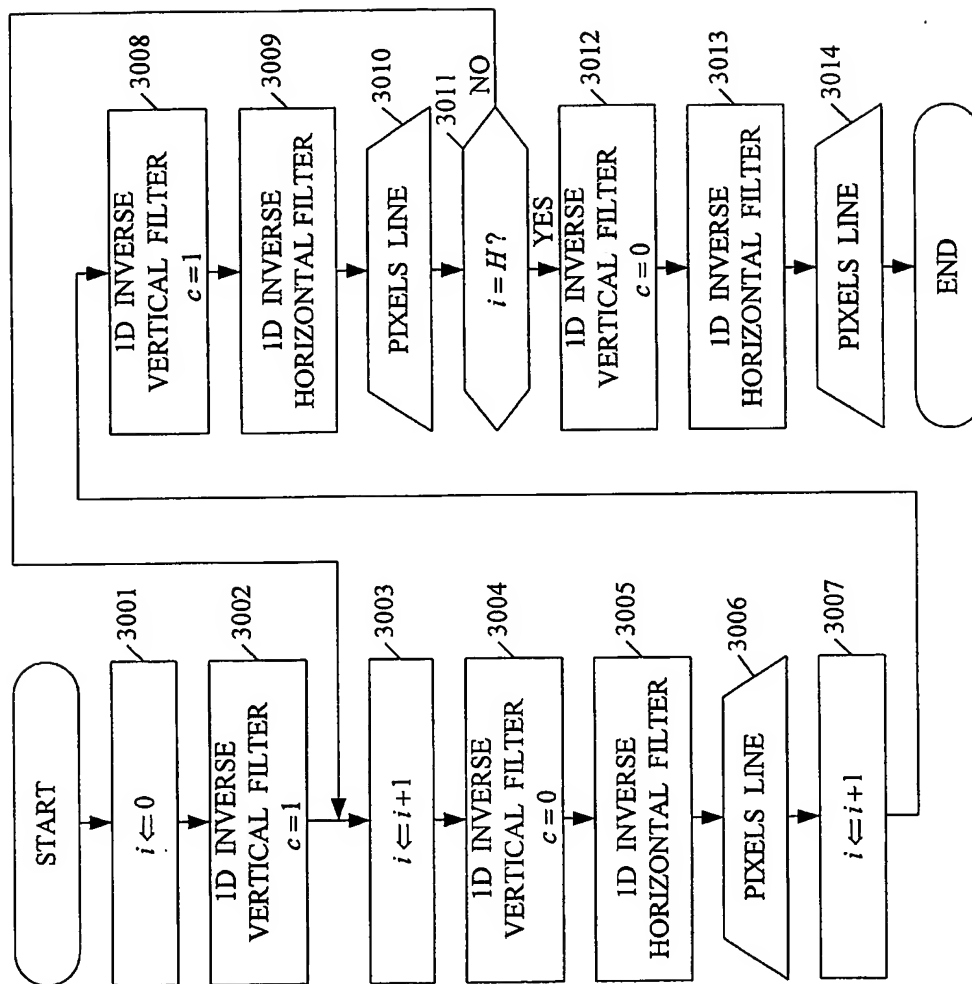


FIG. 30

25/34

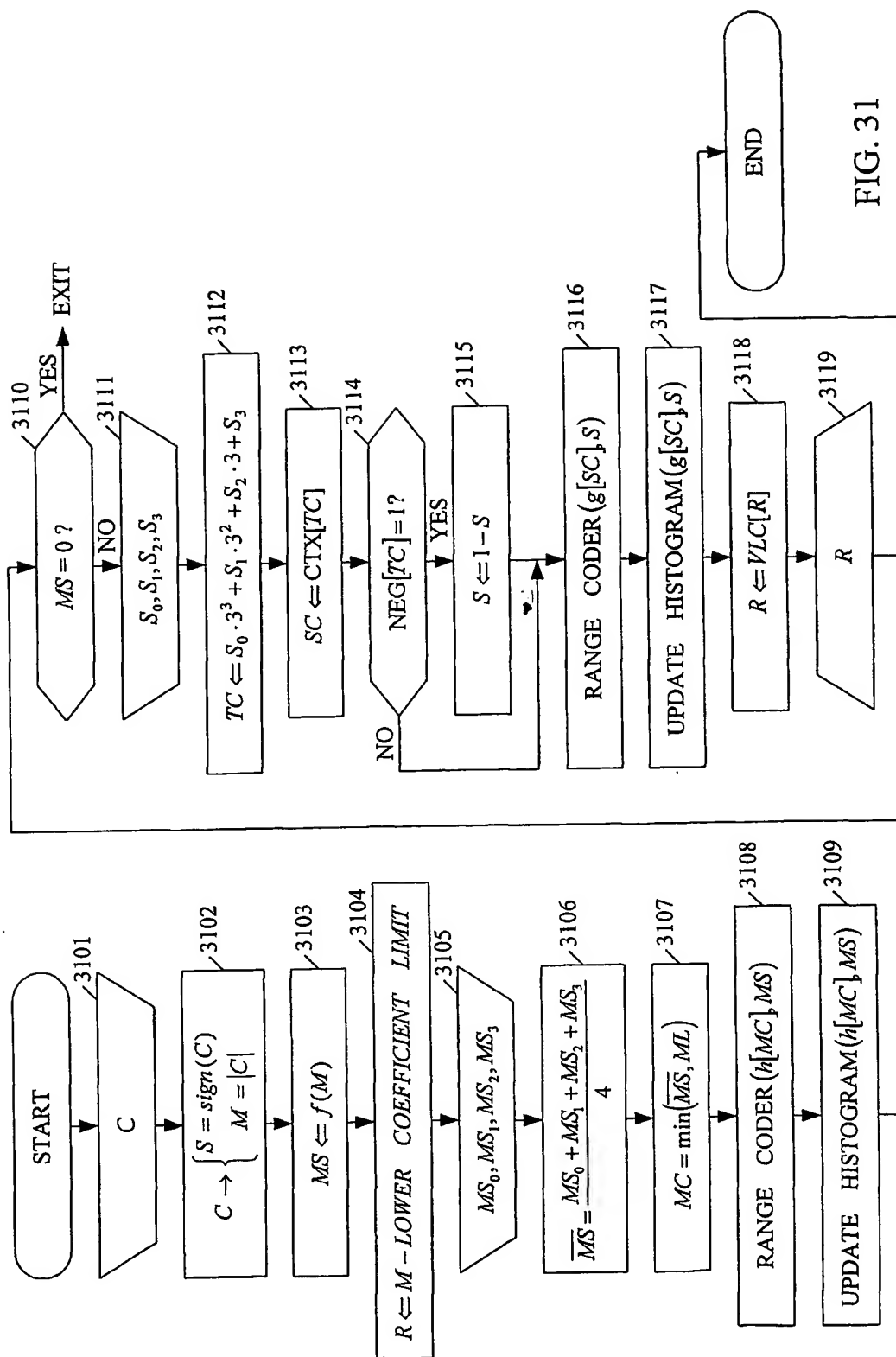


FIG. 31

| | | |
|--------|--------|--------|
| MS_0 | MS_1 | MS_2 |
| MS_3 | MS | |

FIG. 32

| | | |
|-------|-------|-------|
| S_0 | S_1 | S_2 |
| S_3 | S | |

FIG. 33

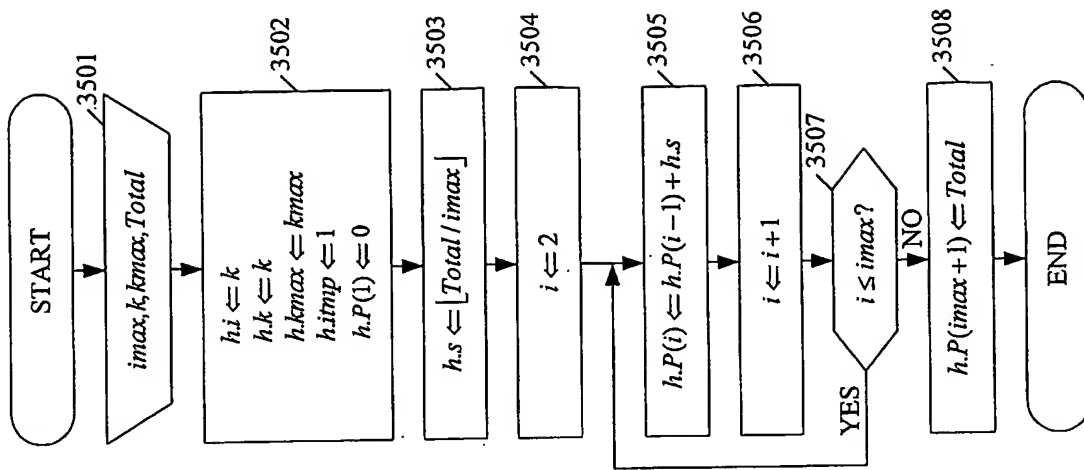


FIG. 35

27/34

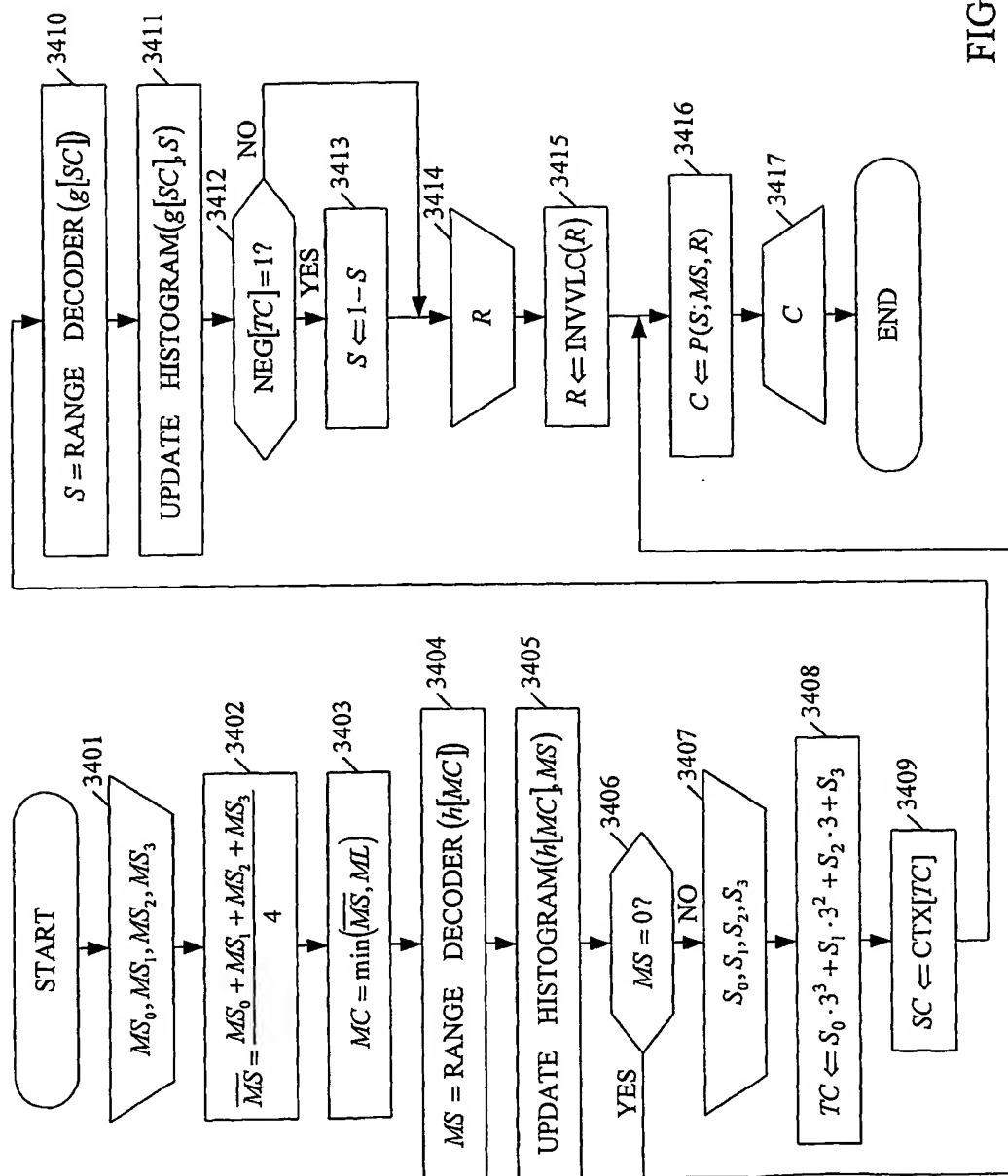


FIG. 34

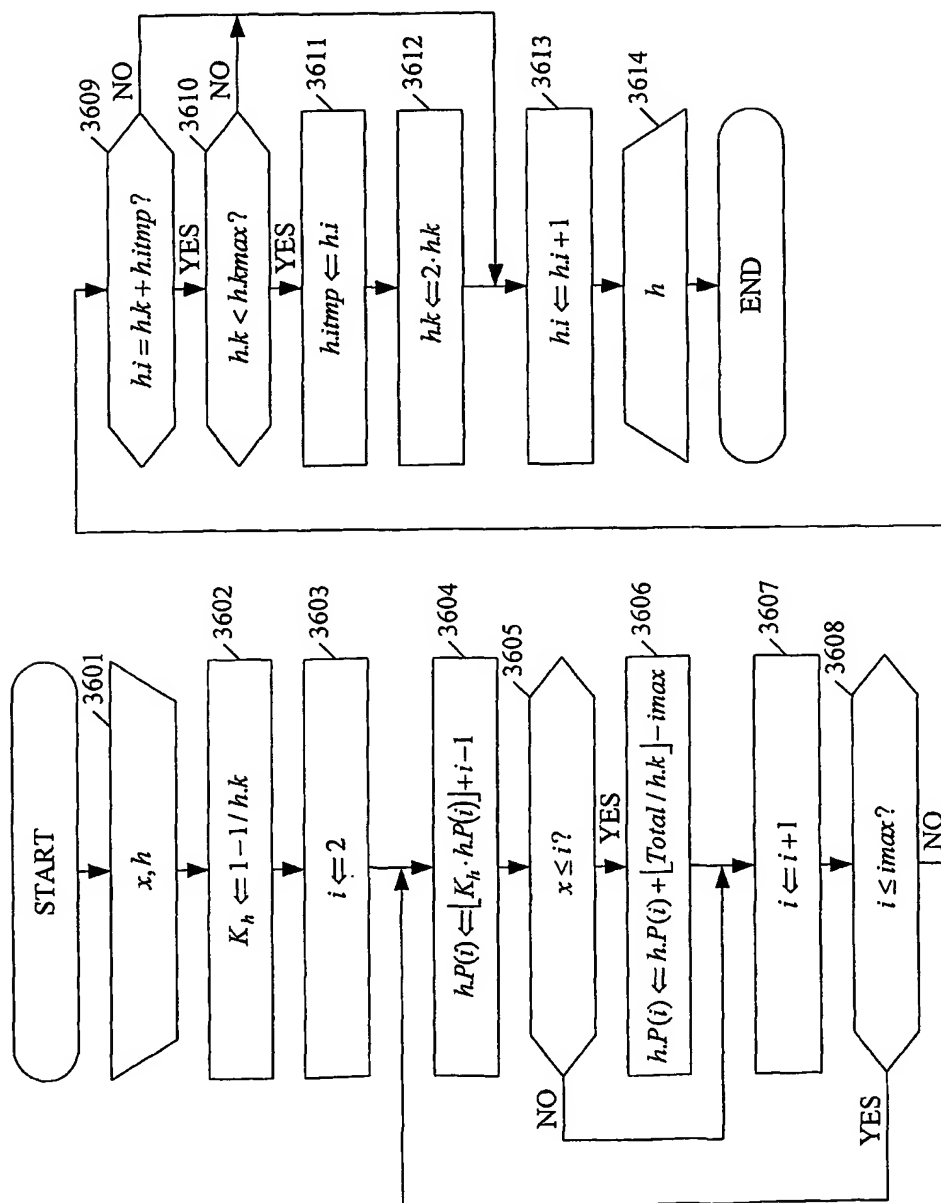


FIG. 36

29/34

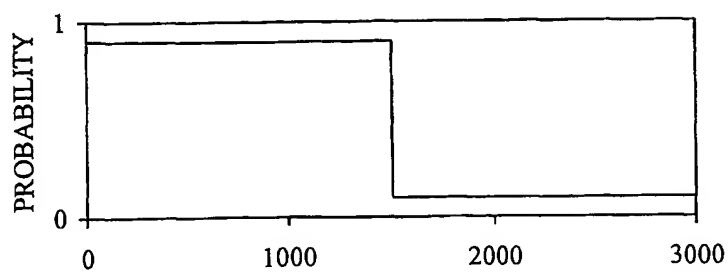


FIG. 37A

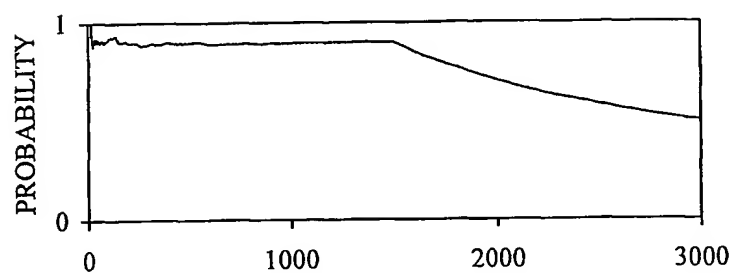


FIG. 37B

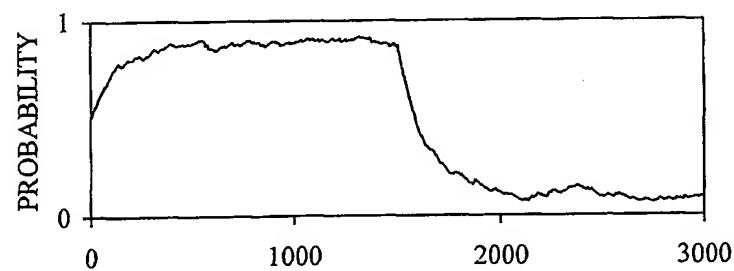


FIG. 37C

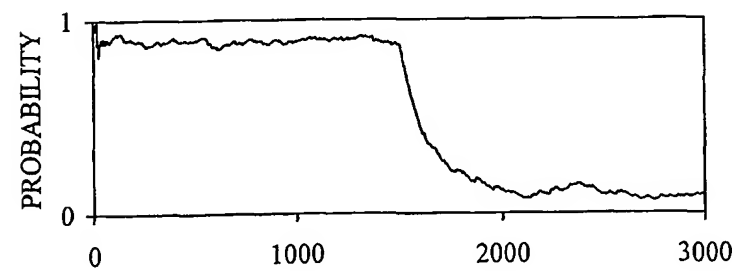


FIG. 37D

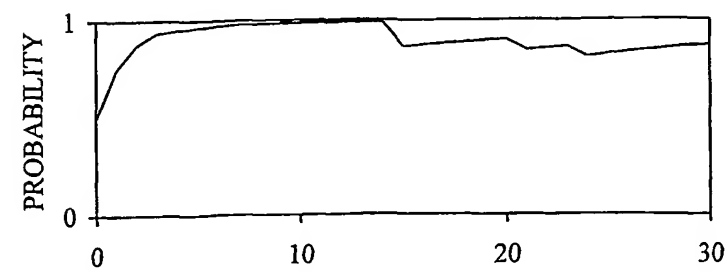


FIG. 37E

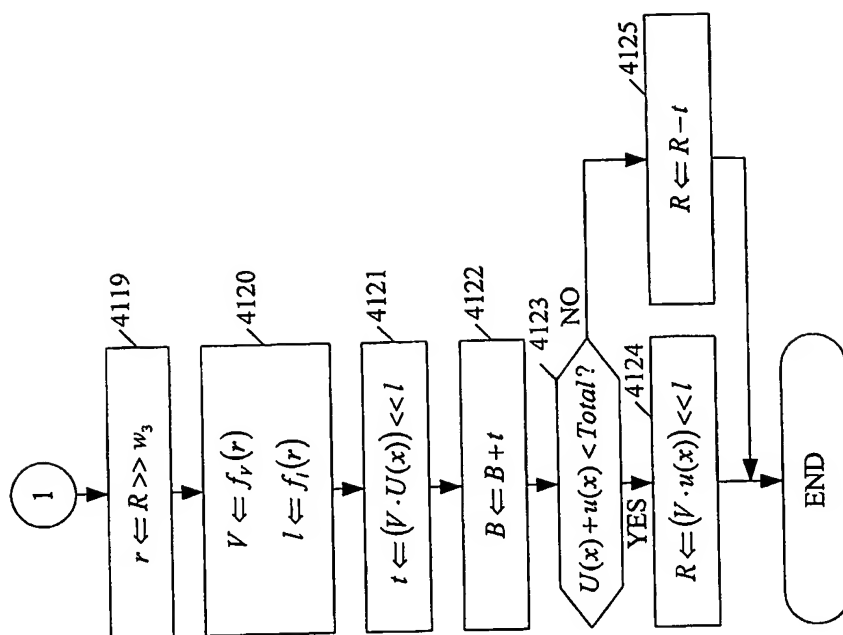


FIG. 41B

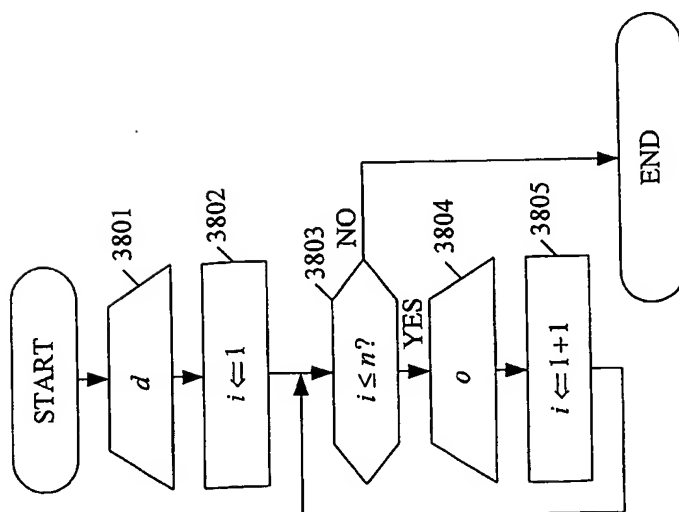


FIG. 38

31/34

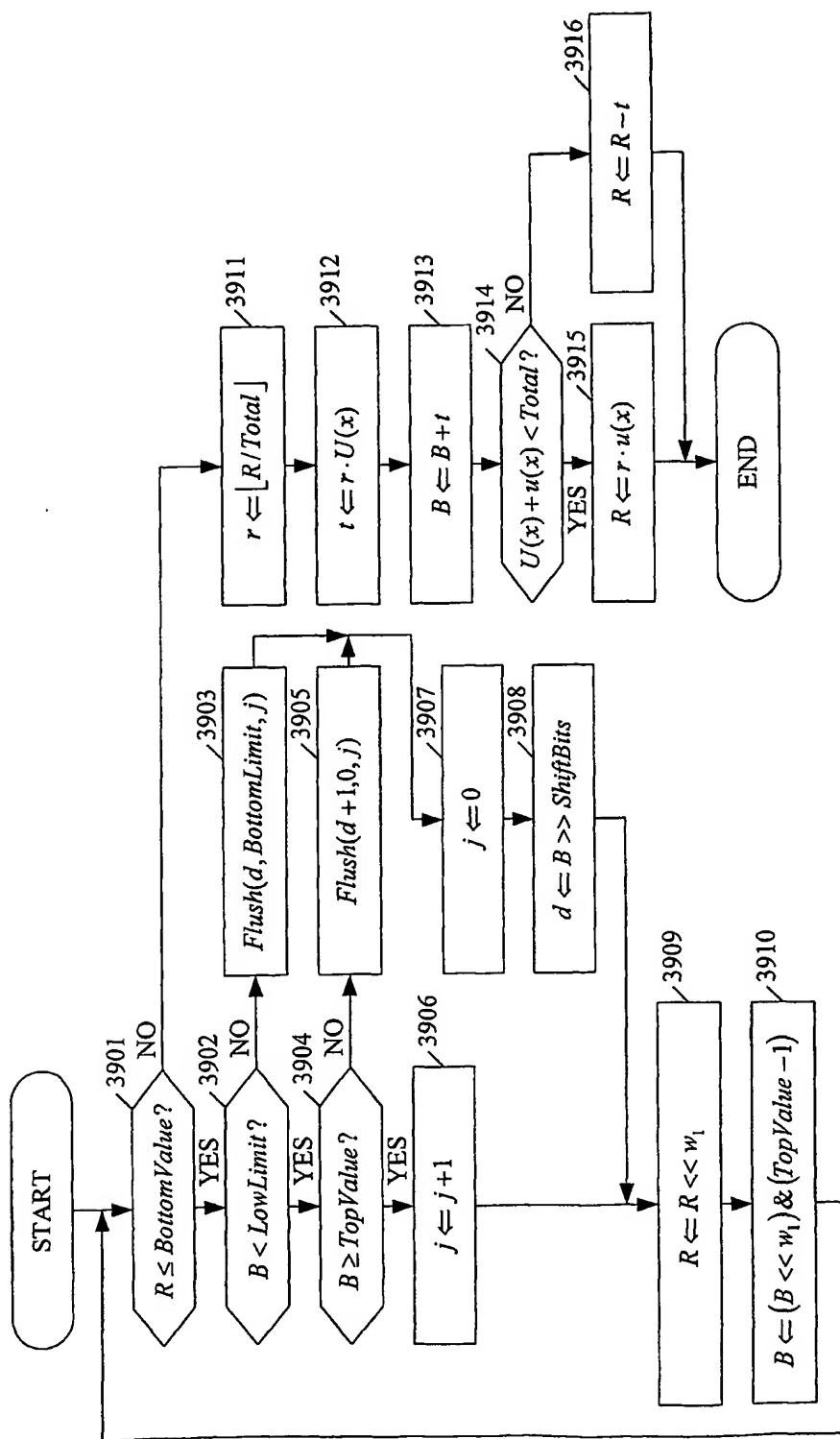


FIG. 39

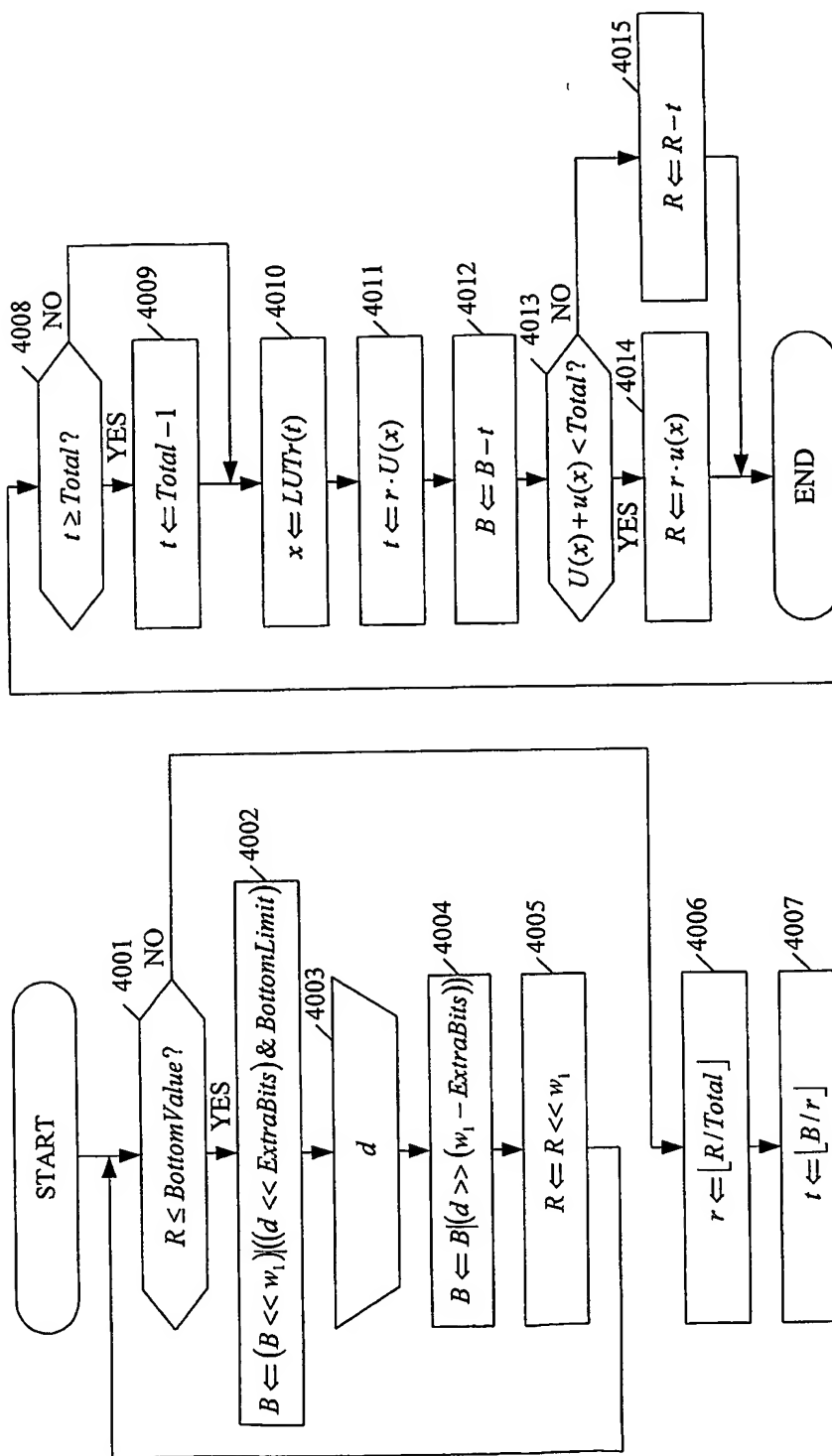


FIG. 40

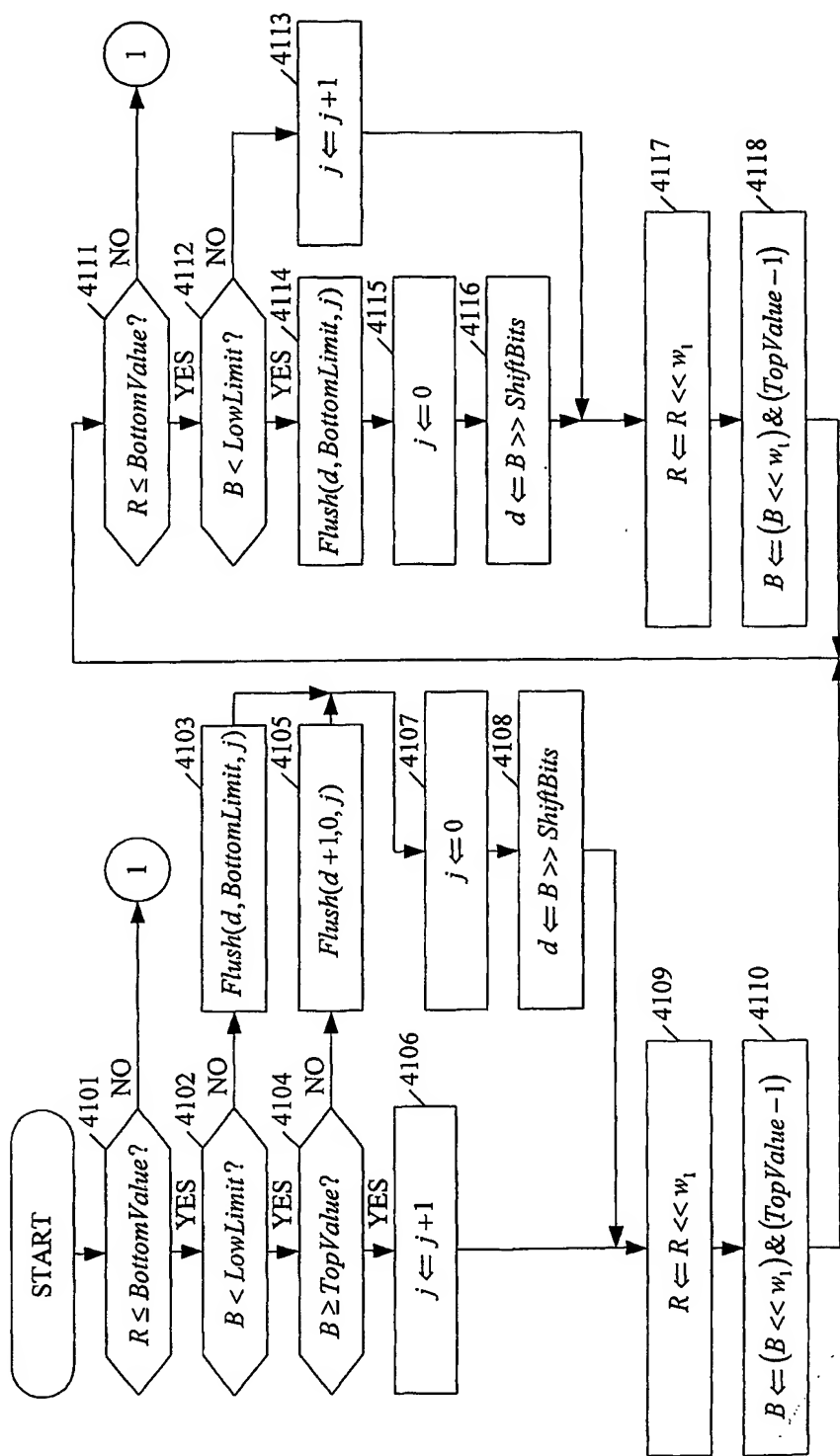


FIG. 41A

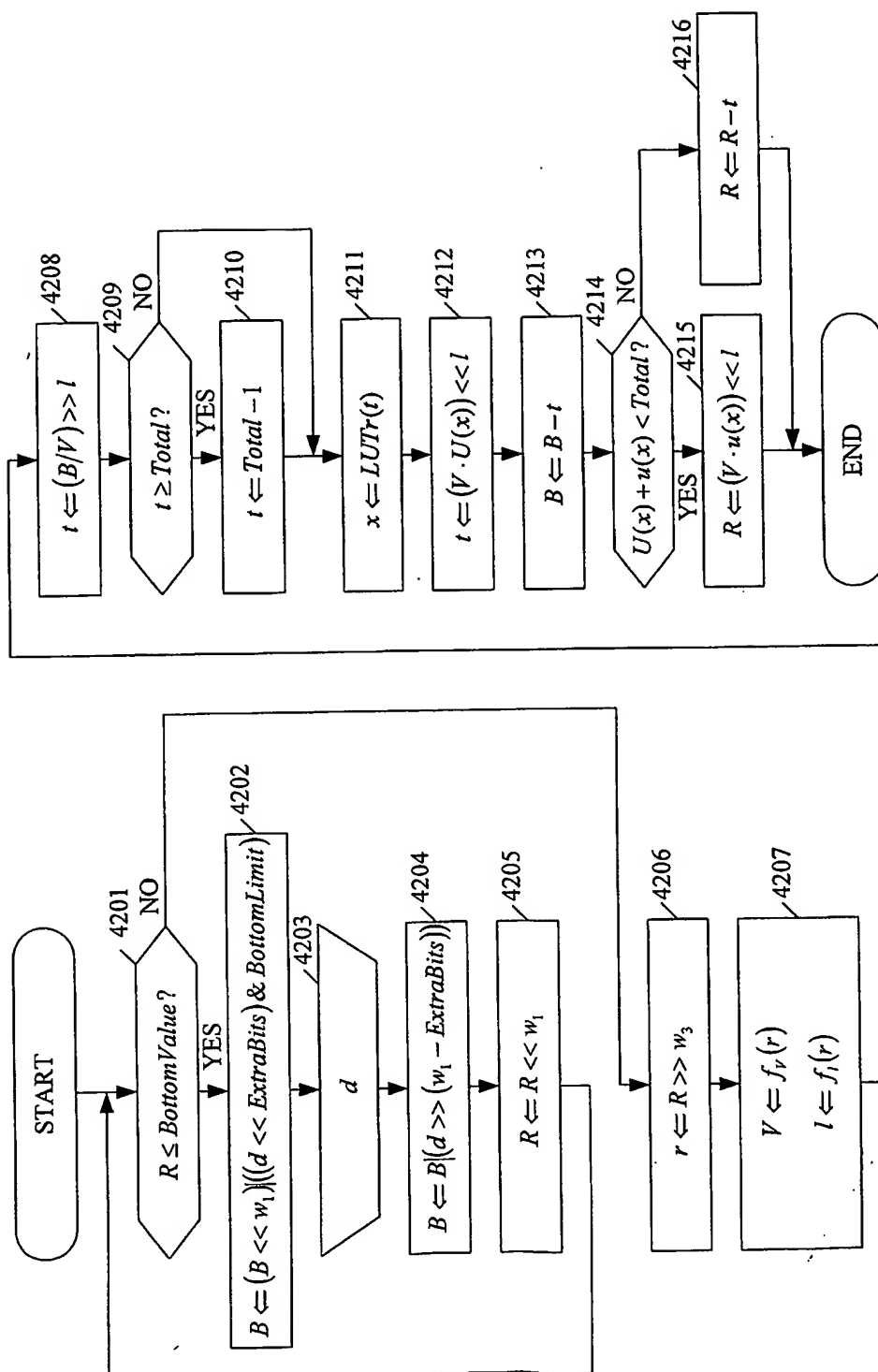


FIG. 42

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant:

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.